TRENDS IN MATHEMATICS

Symbolic-Numeric Computation

Dongming Wang
Lihong Zhi
Editors

Birkhäuser

Symbolic-Numeric Computation

Dongming Wang
Lihong Zhi

Editors

Birkhäuser
Basel · Boston · Berlin

Editors:

Dongming Wang
School of Science
Beihang University
37 Xueyuan Road
Beijing 100083
China

and

Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie - CNRS
8, rue due Capitaine Scott
75015 Paris
France
e-mail: Dongming.Wang@lip6.fr

Lihong Zhi
Key Laboratory of Mathematics Mechanization
Academy of Mathematics and System Sciences
Beijing 100080
China
e-mail: lzhi@mmrc.iss.ac.cn

2000 Mathematical Subject Classification 68W30; 65K10

Library of Congress Control Number: 2006937471

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.ddb.de

ISBN 978-3-7643-7983-4 Birkhäuser Verlag AG, Basel – Boston – Berlin

Printed on acid-free paper produced from chlorine-free pulp. TCF ∞
Printed in Germany
ISBN-10: 3-7643-7983-9 e-ISBN-10: 3-7643-7984-7
ISBN-13: 976-3-7643-7983-4 e-ISBN-13: 978-3-7643-7984-1

9 8 7 6 5 4 3 2 1 www.birkhauser.ch

Preface

This book contains original research articles and tutorial surveys on the interaction and combination of symbolic and numeric computations. It has evolved from the invited and contributed presentations given at the International Workshop on Symbolic-Numeric Computation (SNC 2005 – http://www-calfor.lip6.fr/~wang/SNC2005/) held in Xi'an, China during July 19–21, 2005. Among the 23 chapters in the book, four are contributed by the SNC 2005 invited speakers, Robert M. Corless, Matu-Tarow Noda, Victor Y. Pan (and co-workers), and Hans J. Stetter, 18 were selected by the SNC 2005 Program Committee from 39 submissions with a double-refereeing process, and one additional tutorial survey was submitted under invitation. The collection of articles covers various topics of symbolic and numeric computations:

– GCD computation and factorization for polynomials with inexact coefficients
– Symbolic-numeric methods for solving polynomial systems
– Symbolic-numeric linear algebra and nonlinear algebra
– Resultants and structured matrices for symbolic-numeric computation
– Differential equations for symbolic-numeric computation
– Numeric polynomial algebra and algebraic geometry
– Numeric computation of triangular sets, Gröbner bases, and border bases
– Implementation of symbolic-numeric algorithms
– Applications of symbolic-numeric computation

The growing demand of speed, accuracy, and reliability in scientific and engineering computing has been accelerating the merging of symbolic and numeric computations, two types of computation coexisting in mathematics yet separated in traditional research of mathematical computation. SNC 2005 was organized as a satellite conference of the 2005 International Symposium on Symbolic and Algebraic Computations to facilitate the symbolic-numeric interaction and integration and meanwhile to resume the previous meetings SNAP '96 held in Nice, France and the SNAP mini-symposium at SIAM '98 organized for the same objective of designing and implementing algorithms and software tools for hybrid computation.

The SNC 2005 program featured four invited talks and 27 technical presentations. It provided a lively forum for researchers in the fields of computer algebra and numerical analysis to interact and exchange ideas and views. Informal proceedings of SNC 2005 were printed and distributed at the meeting. The workshop attracted 55 participants from Austria, Canada, China, France, Germany, Japan, and USA.

We acknowledge the generous support received from the National Natural Science Foundation of China, the Key Laboratory of Mathematics Mechanization, Institute of Systems Science, Academy of Mathematics and System Sciences, and the Key Laboratory of Mathematics, Informatics and Behavioral Semantics, School of Science, Beihang University. Our special thanks are due to the members of the SNC 2005 Program Committee and external referees for their expertise and timely help. We thank all the authors for their contributions to SNC 2005 and this volume, and Kaitai Li and his organization team from Xi'an Jiaotong University for their efforts with the local arrangements of SNC 2005. We are also grateful to Thomas Hempfling and Stefan Göller at Birkhäuser, Basel for their support and cooperation on the publication of this volume.

Beijing
October 2006

Dongming Wang
Lihong Zhi

Contents

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 1–15

On a Generalized Companion Matrix Pencil for Matrix Polynomials Expressed in the Lagrange Basis

Robert M. Corless

In memoriam: Karin Gatermann (1961–2005)

Abstract. Experimental observations of univariate rootfinding by generalized companion matrix pencils expressed in the Lagrange basis show that the method can sometimes be numerically stable. It has recently been proved that a new rootfinding condition number, defined for points on a set containing the interpolation points, is never larger than the rootfinding condition number for the Bernstein polynomial (which is itself optimally small in a certain sense); and computation shows that sometimes it can be much smaller. These results extend to the matrix polynomial case, although in this case we are not computing polynomial roots but rather 'polynomial eigenvalues' (sometimes known as 'latent roots'), *i.e.* finding the values of x where the matrix polynomial is singular. This paper gives two theorems explaining part of the influence of the geometry of the interpolation nodes on the conditioning of the rootfinding and eigenvalue problems.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 65F99; Tertiary 65D99.

Keywords. Lagrange basis, companion matrix, polynomial eigenvalue.

1. Introduction

There are two parallel threads of recent research into polynomial computation using alternative bases, other than the power basis. The motivation for both threads is that *conversion between bases can be unstable*, and the instability increases with the degree [17]. These threads are of interest both to the computer algebra

This work was supported in part by a grant from the Natural Science and Engineering Research Council of Canada.

community and to the numerical analysis community because both threads involve hybrid symbolic-numeric computation.

One thread, exemplified by the papers [7, 14, 15, 30, 32] investigates polynomial computation via the Bernstein basis, which is well-suited to applications in computer-aided geometric design. The paper [31] compares the average-case behaviour of monomial and Bernstein bases and shows that Bernstein bases are better. Further, the papers [13, 22] prove that in a certain sense Bernstein bases are optimal both for evaluation and for rootfinding.

The other thread, exemplified by the papers [1, 5, 9, 27], investigates computation with polynomials expressed in the Lagrange basis, or in other words directly by values. Related works include [16, 28], which use the Lagrange basis as an intermediate step in the computation and analysis of polynomial roots.

The condition number of the rootfinding problem for scalar polynomials expressed in the Lagrange basis was studied in [11], which showed that while Bernstein bases are optimal in the class of bases nonnegative on the interval $[0,1]$, Lagrange bases can sometimes be better (although they can be negative on the interval). See Sect. 2.5 for a summary of those results. Another result of that paper was that the rootfinding condition number can sometimes be *arbitrarily* better, in the case when a node is arbitrarily near to a root.

Recent papers by Berrut and Trefethen [6] and Higham [19] show that working directly in the Lagrange basis is both numerically stable and efficient, more stable and efficient than had been heretofore credited widely in the numerical analysis community. These recent results strengthen the motivation for examining algorithms for direct manipulation of polynomials by values. We use the notation of [6, 19]. They show that one may write the (scalar) polynomial $\mathbf{P}(x)$ that takes on the values $\mathbf{P_k}$ at the distinct nodes x_k in either of the two *barycentric forms*:

$$\mathbf{P}(x) = \ell(x) \sum_{k=0}^{n} \frac{w_k}{x - x_k} \mathbf{P_k} \tag{1.1}$$

or

$$\mathbf{P}(x) = \left(\sum_{k=0}^{n} \frac{w_k \mathbf{I}}{x - x_k} \right)^{-1} \sum_{k=0}^{n} \frac{w_k}{x - x_k} \mathbf{P_k} \tag{1.2}$$

where $\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$, and where the $w_k = 1/\prod_{j \neq k}(x_k - x_j)$ are the (scalar) normalization factors, called the *barycentric weights* in [6], of the Lagrange polynomials $L_k(x) = w_k \prod_{j \neq k}(x - x_j)$.

Extension of the first form, (1.1), to the matrix polynomial case is straightforward: the formula holds componentwise. To see that the second form is valid, interpolate the constants 1 and 0 on these nodes to find that

$$\mathbf{I} = \ell(x) \sum_{k=0}^{n} \frac{w_k}{x - x_k} \mathbf{I}$$

and hence that $\ell(x)\mathbf{I} = \left(\sum_{k=0}^{n} \frac{w_k \mathbf{I}}{x - x_k} \right)^{-1}$. This is equivalent to division by a scalar.

The works [6, 19] show that these formulae are *numerically stable*, in that the values obtained by evaluating these formulae for a given numerical x are the exact values of polynomials that nearly go through the given data $\mathbf{P_k}$. More particularly, they show this in the scalar case, but this just extends componentwise to matrix polynomials.

All the usual caveats about the use of high-degree interpolating polynomials and their sensitivity to changes or errors in the data must be considered in any given application. This observation motivates the present paper, where we explore how the placement of nodes affects the conditioning (sensitivity) of the polynomial eigenvalue problem for a matrix polynomial given in the Lagrange basis.

This current paper takes the results of [11] further, extending the results to the matrix polynomial case, and investigating the effect of the geometry of the interpolation nodes on the conditioning of the problem.

2. Summary of Previous Work

2.1. A Generalized Companion Matrix in the Lagrange Basis

Following [9], the *generalized companion matrix* of the polynomial given by its values is defined below. If the values of the (matrix) polynomial at $x = x_0$, $x = x_1$, . . ., and x_n are (the matrices) $\mathbf{P_0}$, $\mathbf{P_1}$, . . . and $\mathbf{P_n}$, then the generalized companion matrix pencil is

$$\mathbf{C_0} = \begin{bmatrix} x_0\mathbf{I} & & & & \mathbf{P_0} \\ & x_1\mathbf{I} & & & \mathbf{P_1} \\ & & \ddots & & \vdots \\ & & & x_n\mathbf{I} & \mathbf{P_n} \\ w_0\mathbf{I} & w_1\mathbf{I} & \cdots & w_n\mathbf{I} & 0 \end{bmatrix} =: \begin{bmatrix} \mathbf{X} & \pi \\ \mathbf{w}^T & \mathbf{0} \end{bmatrix} \tag{2.1}$$

where $\mathbf{X} := \operatorname{diag}(x_0, x_1, \ldots, x_n)$, π is the last s columns and first $(n+1)s$ rows of C_0, and $\mathbf{w}^T$ is the last s rows and first $(n+1)s$ columns of C_0, and $C_1 = \operatorname{diag}(\mathbf{I}, \mathbf{I}, \ldots, \mathbf{I}, \mathbf{0})$ where the blocks $\mathbf{I}$ are conformal with the $s \times s$ square blocks $\mathbf{P_k}$. Then we have

Theorem 2.1.

$$\det(x\mathbf{C_1} - \mathbf{C_0}) = \det \mathbf{P}(x)$$

and $\deg \det \mathbf{P}(x) \leq ns$.

Proof. See [9], or use the Schur complement. □

In the $s = 1$ (scalar) case, we will write $p(x)$ instead of $\mathbf{P}(x)$. In that case, the eigenvalues of the pencil $(\mathbf{C_0}, \mathbf{C_1})$ are, aside from an extraneous double root at infinity, exactly the roots of $p(x)$. Notice that the Lagrange polynomial is not converted to monomial form, or indeed even formed explicitly, though the barycentric weights w_k *are* formed.

2.1.1. Scaling. Clearly, we may multiply all the $\mathbf{P_k}$ by any nonsingular matrix whatever without changing the generalized eigenvalues, because this simply multiplies the matrix polynomial by a constant matrix. This can be shown to imply that we may also multiply the final s rows of C_0 by any nonsingular matrix also without changing the generalized eigenvalues.

Finally, we may scale all x_k by a constant, say $X = \max |x_k|$ and put $\xi_k = x_k/X$. This induces a scaling of the barycentric weights by X^n. The polynomial eigenvalues of the scaled C_0, C_1, say μ_{ij}, are related to the original polynomial eigenvalues λ_{ij} by $\mu_{ij} = \lambda_{ij}/X$.

Remark. We will assume henceforth that the nodes x_k and the values $\mathbf{P_k}$ are scaled in a fashion suitable for computation (e.g. by dividing by a constant so that the maximum norm $\|\mathbf{P}_m\| = 1$ and by dividing all the w_k by their maximum) *by a person knowledgable about the particular application context.* We do not yet know enough about this to recommend any particular automatic scaling in what follows, though we do recommend that the person doing the computation pay attention to this issue. Use of unscaled data corresponds to using non-monic polynomials in the monomial case.

Remark 2. Experiments with the companion matrix pencil for the scaled Wilkinson polynomial $p(x) = \prod_{k=1}^{20}(x - k/21)$ show that a proper scaling[1] such as discussed above is crucial for accurate results *if the standard QZ iteration is used to solve the polynomial eigenvalue problem via the linearization to C_0, C_1.* That is, simply forming C_0 and C_1 and passing these unbalanced to an iterative eigenvalue problem solver such as QZ iteration does not give good results, because the balancing necessary is not permissible (in general) for a generic matrix pencil and thus is not part of any standard implementation of the QZ iteration. Since the generic QZ iteration is of cost $O(n^3s^3)$, as either n or s goes to infinity with the other fixed, and does not take advantage of structure, we would hope to do better in any case.

In some detail, the example was to find the roots of a 2 by 2 matrix polynomial formed from the scaled Wilkinson polynomial $p(x) = \prod_{k=1}^{20}(x - k/21)$. We chose nodes randomly near the interval $[0, 1]$, by the Matlab command `nodes = linspace(0,1,21) + (rand(1,21)-0.5)*(.007+.007*sqrt(-1))/21;`, and then set up the polynomial values at the nodes by the following iteration:

```
rts = (1:20)/21
yd = nodes - rts(1)
for i=2:20,
  yd = yd .* (nodes-rts(i));
end;
```

[1]I will call this 'balancing' in the future, but it is different from the standard 'balancing' of matrices for stability in computing eigenproblems, where a diagonal similarity transformation is used to make the row and column norms more-or-less equal. Here we have more flexibility.

This evaluates $p(x)$ accurately at each node, using the product form. We then made a two by two system by using the Kroenecker product: `y = kron(yd,eye(2))`. We then set up the generalized companion matrix pencil, and balanced it:

```
[c0,c1,w] = gcmp( nodes, y' )
% Must balance c0 for good numerical results!
c0( :, 43:44 ) = c0( :, 43:44 )/norm( c0(:,43:44) );
c0( 43:44, : ) = c0( 43:44, : )/norm( c0(43:44,:));
```

The computed eigenvalues of this companion matrix pencil, *i.e.* the eigenvalues of the linearization of the matrix polynomial, were different from the true latent roots by no more than $7\cdot 10^{-4}$, and are plotted in Fig. 1 together with the roots and with the roots obtained by interpolating the y values (forming the matrix polynomial in the monomial basis) and calling `polyeig`. The errors in the interpolate-then-solve-with-`polyeig` case were nearly 100 times larger, as is visible in the Figure.

Without balancing, the errors in the eigenvalues of the pencil C_0, C_1 were also smaller than the errors in the interpolation case, but only accidentally so: the nodes themselves are close to the roots, and the computed eigenvalues of the unbalanced pencil were just the nodes.

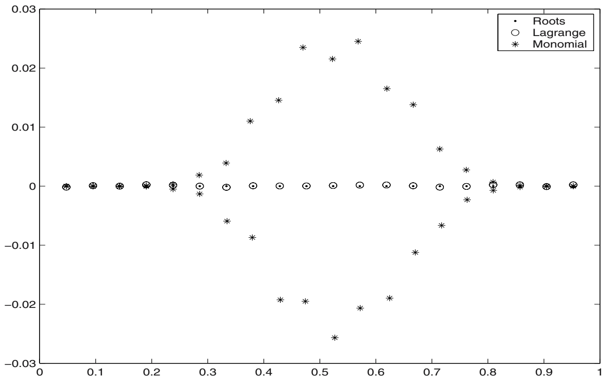


FIGURE 1. Roots of the scaled Wilkinson polynomial, embedded in a two by two matrix polynomial, as computed by the linearization (generalized companion matrix) of the Lagrange form, and as computed by first interpolating and then using MATLAB's built-in `polyeig`. The errors in the interpolate-then-solve method are nearly 100 times larger than the direct method.

2.2. Eigenvectors for Simple Finite Eigenvalues

2.2.1. Barycentric Form. If r is not a node but is an eigenvalue of the matrix polynomial $\mathbf{P}(x)$ and $\mathbf{v}$ ($\mathbf{w}^T$) is a corresponding right (left) null vector of $\mathbf{P}(r)$, then the eigenvectors of the generalized companion matrix pencil (2.1) corresponding to the simple finite eigenvalue r are

$$R = \begin{bmatrix} \mathbf{P_0}\mathbf{v}/(r-x_0) \\ \mathbf{P_1}\mathbf{v}/(r-x_1) \\ \vdots \\ \mathbf{P_n}\mathbf{v}/(r-x_n) \\ \mathbf{v} \end{bmatrix} \tag{2.2}$$

and on the left

$$L = \left[w_0/(r-x_0)\,\mathbf{w}^t, w_1/(r-x_1)\,\mathbf{w}^t, \ldots, w_n/(r-x_n)\mathbf{w}^t, \mathbf{w}^t \right] . \tag{2.3}$$

2.3. Generalized Eigenvectors of Multiple Finite Eigenvalues

Letting an eigenvalue r_1 approach another eigenvalue r_2 and examining the linear combinations of eigenvectors $(R_2-R_1)/(r_2-r_1)$ and $(L_2-L_1)/(r_2-r_1)$, with the obvious interpretation of symbols, shows that the right eigenspace of a multiple eigenvalue is spanned by R, $D(R)(r)$, and indeed all derivatives up to order $m-1$ where m is the order of multiplicity of the eigenvector. This is exactly what would be expected, and a similar result holds for the left eigenspace.

2.4. The Reducing Subspace at Infinity

Consider the scalar case, $s=1$.

Theorem 2.2. *If $p^{(n)} \not\equiv 0$ then there is a double eigenvalue at infinity, with only one eigenvector $[0,0,\ldots,0,1]^T$. Another vector in the reducing subspace at infinity is $[p_0,p_1,\ldots,p_n,0]^T$.*

The case $s>1$ is similar:

Theorem 2.3. *If $\mathbf{P}^{(n)}$ is not singular, then there are s double eigenvalues at infinity, with eigenvectors $[\mathbf{0},\mathbf{0},\ldots,\mathbf{0},\mathbf{e}_k^T]$ for $1 \le k \le s$. Another s vectors in the reducing subspace at infinity are the columns of $[\mathbf{P}_0,\mathbf{P}_1,\ldots,\mathbf{P}_n,\mathbf{0}]$.*

For a proof, see [10].

Remark. These 'intrinsic' roots at infinity do not usually bother our computations. If we use the RQI method (or any other numerical method that undoes the linearization, by taking account of the structure) then these infinite roots are automatically avoided. However, we see in Sect. 4.3 that if the degree of $\mathbf{P}(x)$ is actually less than n, then there are still more spurious roots at infinity; and the higher the multiplicity, the more chance that they may affect the computation of the finite roots.

2.5. Conditioning

The *conditioning* of a problem measures sensitivity of the solution to changes in the problem data. This notion is extremely useful in applied mathematics and in numerical analysis, because it can also be used to estimate the sensitivity of the solution to changes in the problem *formulation* and, by virtue of *backward error* *analysis*, the sensitivity of the problem to numerical errors. There are hierarchies of condition numbers for several problems, and in [22] we find an analogue of the *structured* condition number of linear algebra (see [18] for an overview) defined and used for evaluation of polynomials; the paper [13] shows that dividing an evaluation condition number by $|p'(r)|$ gives a rootfinding condition number. We summarize the analogues in the context of Lagrange bases, here.

Let U be a finite-dimensional vector space of functions defined on $\Omega \in \mathbb{C}^s$ and let $b = (\phi_0, \phi_1, \ldots, \phi_n)$ be a basis for U. Let $T \in \Omega$ be a (possibly finite) set which we will use to characterize nonnegativity of the basis: both [13] and [22] take $T = \Omega$ but we will occasionally take $T \subset \Omega$ to be a set containing the interpolation points. If $f \in U$ has simple finite roots, and the expansion $f(x) = \sum_{i=0}^{n} c_i\phi_i(x)$, we consider the relatively perturbed function $g = \sum_{i=0}^{n}(c_i + \delta_i c_i)\phi_i(x)$ and look at the differences between the simple roots of g and those of the unperturbed f. Similar considerations hold for the values of g compared to the values of f.

$$C_b(f, x) := \frac{1}{|f'(x)|}\sum_{i=0}^{n}|c_i\phi_i(x)| \tag{2.4}$$

$$\mathrm{cond}(b; f, x) := \frac{C_b(f, x)}{\|f\|_\infty} \tag{2.5}$$

$$\mathrm{cond}_T(b, f) := \sup_{x\in T}\mathrm{cond}(b; f, x) \tag{2.6}$$

With $\varepsilon = \|\delta c\|_\infty$ we have (asymptotically as $\varepsilon \to 0$)

$$|r - \mathrm{RootOf}(g, x = r)| = C_b(f, r)\varepsilon + O(\varepsilon^2)$$

where $\mathrm{RootOf}(g, x = r)$ means the root of g closest to r.

Remark. A similar definition holds for the matrix polynomial case, but the formulae are more complicated, involving the left and right null vectors of $\mathbf{P}(r)$. For example, the conditioning of a simple eigenvalue r with left null vector $\mathbf{w}^T$ and right null vector $\mathbf{v}$ is given by the formula

$$\delta_r \approx \frac{\mathbf{w}^T\Delta\mathbf{P}(r)\mathbf{v}}{\mathbf{w}^T\mathbf{P}'(r)\mathbf{v}}\,. \tag{2.7}$$

Here the eigenvalue $r + \delta_r$ of $\mathbf{P}(x) + \Delta\mathbf{P}(x)$ is related to the eigenvalue r of $\mathbf{P}(x)$.

3. Numerical Methods for Computing the Eigenvalues

Without efficient and stable methods for computing generalized eigenvalues of matrix pencils, this new companion matrix pencil would be a curiosity only. In this section I make some remarks on numerical methods.

3.1. Use of Existing General-Purpose Methods

The simplest approach is just to call existing routines in Maple or Matlab or whatever libraries the user has handy for the computation of generalized eigenvalues of matrix pencils. This is *so successful* an approach that the motivation for searching for better methods is considerably lessened. Indeed, all previous examples of computation of roots in this method, including the over 400,000 roots computed in [9], were computed by calling standard routines. However, there are three pertinent observations:

1. For standard methods, such as QR iteration for the ordinary eigenvalue problem or QZ for the generalized eigenproblem, the cost in storage is $O(n^2)$ and the cost of the arithmetic is $O(n^3)$ flops (if that matters: on some parallel machines, for example, the claim is often made that floating-point arithmetic is free, taking place as it does in the shadow of memory accesses). In theory, the storage cost for these special matrices should be $O(n)$ and the arithmetic cost could (at first glance) be expected to be $O(n^2)$, and so one would expect that special-purpose methods would be cheaper. One would also expect that in certain applications this cost savings of a factor of n would actually matter.
2. The proof that 'finding roots by way of standard methods to compute eigenvalues of companion matrices' is numerically stable [12, 29] is published only for companion matrices in the monomial basis, and apparently, in a Ph.D. thesis by Gudbjorn Jónsson, for the Bernstein basis (see [21] for a citation). It seems clear that the proofs will carry over to this new companion matrix pencil, but the details remain to be worked out. In particular, the observations about balancing (more special-purpose balancing than can be done for general matrix pencils) by scaling the x_k, the $\mathbf{P}_k$, and the barycentric weights may play a significant role.
3. Some older methods, such as Rayleigh quotient iteration, have natural interpretations in terms of matrix-vector products and can easily be cast in a way that requires only $O(n)$ storage and $O(n^2)$ arithmetic cost; but possible problems with deflation, basins of attraction for initial guesses, etc remain to be investigated.

3.2. Rayleigh Quotient Iteration

Amirhossein Amiraslani, Dhavide Aruliah and I have implemented two variations of Rayleigh quotient iteration for the computation of eigenvalues and eigenvectors of these generalized companion matrix pencils. These will be reported on more fully elsewhere [3], but, in brief, we attain $O(ns^2)$ storage use and $O(n^2s^2 + s^3)$ arithmetic cost in both variations; however, the constrained case requires initial guesses to be near the convex hull of the interpolation nodes, and thus is inferior to the slightly-more-expensive-per-iteration unconstrained method, which experimentally shows convergence almost everywhere.

3.3. General Methods for Quasi-Separable Matrices

Dario Bini and co-workers have discovered and implemented a QR-based algorithm that *preserves the structure* of a class of structured matrices, including the type of matrix pencils used in this paper [8]. This is a very significant development, and in particular their methods ought to be easily extendable to the matrix polynomial case, where we would get, again, $O(ns^2)$ storage cost and $O(n^2 s + s^3)$ flops, all the while maintaining numerical stability.

3.4. Weierstrass Iteration

Both Rayleigh quotient iteration and the quasi-separable matrix methods alluded to previously use the structure of the matrix pencil but only indirectly use the fact that the eigenproblem is that of a matrix polynomial expressed in the Lagrange basis. One wonders if the use of Weierstrass iteration for simultaneous approximation of all zeros [20, 26] might be a viable, stable, efficient alternative. Weierstrass iteration in the scalar univariate case may be expressed as

$$\overline{z_j} = z_j - \frac{P(z_j)}{\|P\| Q'(z_j)}\,, 1 \le j \le n \tag{3.1}$$

where $Q(z) = \alpha(z - z_1)(z - z_2)\cdots(z - z_n)$ and α is chosen so that $\|Q\| = 1$. Each iteration thus requires the evaluation of P at each z_j, and the evaluation of the derivatives of Q at each z_j, which may be carried out by the differentiation matrix method of [5]. Relevant works include [16] and [24].

4. The Influence of Geometry

4.1. Experimental Results

By direct computation on many examples we have noticed that roots inside a region defined by the interpolation nodes tend to be well-conditioned, whilst roots even a short distance outside this region (think of the convex hull) can be badly conditioned. A thorough study of the average case conditioning of roots, as done by Joab Winkler for the Bernstein basis case [32], would be welcome.

As a first attempt at a systematic exploration, we consider scalar polynomials with values either $+1$ or -1. For low numbers of interpolation nodes, an exhaustive study can be carried out. Note that the constant polynomial has no roots, and that the polynomial with values $-p_k$ has the same roots as the polynomial with values p_k, and therefore we may choose without loss of generality one node, say z_0, to have the value $+1$, and insist that one of z_1, z_2, …, z_n have the value -1. This gives $n2^{n-1}$ different sets of roots of polynomials defined on $n + 1$ nodes.

We have computed all the roots for $n + 1 = 15$ in Maple, using the built-in generalized eigenvalue solver, on a 2003 laptop. Remember, such a computation is guaranteed to be at least a factor of 15 slower than a special-purpose method, but even so computation was successful in less than a day, producing an intelligible plot. See [9].

4.2. The Condition Number of a Faraway Root

We have experimentally observed that roots far away from the interpolation nodes are poorly conditioned. This can be explained analytically, as follows. Suppose we consider a family of polynomials with a root $z = Rr_1$, and $n-1$ other, fixed, roots at r_2, r_3, …, r_n, but scaled so that $p(x_k) = O(1)$ as $R \to \infty$. That is, put

$$p(x) = c(\frac{x}{R} - r_1)(x - r_2)\cdots(x - r_n)\,,$$

for some constant c. We now consider the condition number for relative perturbations at the nodes x_k to the values p_k, that is, changing them to $p_k(1 + \delta_k)$ which is

$$C = \frac{1}{|p'(z)|}\sum_{k=0}^{n} |p_k||L_k(z)|\,.$$

Since $|p'(z)| = |p'(Rr_1)| = O(R^{n-2})$ as $R \to \infty$, and while the sum without the absolute values is zero (z is a root) each Lagrange polynomial becomes large for large R, ie:

$$L_k(z) = \frac{(z - x_0)(z - x_1)\cdots(z - x_{k-1})(z - x_{k+1})\cdots(z - x_n)}{(x_k - x_0)(x_k - x_1)\cdots(x_k - x_n)}$$

and as $R \to \infty$ this clearly grows as $O(R^n)$. Since there are $n + 1$ terms in the sum, we have proved:

Theorem 4.1. *For $n + 1$ fixed interpolation nodes, a distant root of a polynomial ($O(R)$ distant from all interpolation nodes) has a condition number that grows at least as fast as $O((n + 1)R^2)$.*

Proof. The above computation gives the exact condition number for one family of polynomials with a simple large root, thus providing a lower bound on the condition number. $\square$

Remark. The condition number estimates the effect of small *relative* changes in the values of the polynomial at the nodes, but gives *absolute* changes in the root. The above computation shows that there exist polynomials with large roots that each have condition number approximately proportional to the square of the magnitude of the root; so a relative change in the coefficients of $O(\varepsilon)$ *may* result in an absolute change in the large root of $O((n + 1)R^2\varepsilon)$, or a *relative* change in the root of $O((n + 1)R\varepsilon)$. By restricting ourselves to a problem with one single, simple large root, we have only established a lower bound on how bad things can be.

4.3. Perturbing Roots at Infinity

A subtly different issue is the possibility of roots at infinity, caused for example by oversampling. This is different from the previous case because we will be perturbing to or from a *multiple* root, not a simple root. If $p(x)$ has exact degree d, and we interpolate it on $n + 1 > d + 1$ nodes, then rounding errors (or other interpolation errors, such as measurement errors) will give us computed values $\hat{p}_k = p(x_k) + e_k$ for some errors e_k. Exact interpolation of these $n+1$ values will give a polynomial of degree $n > d$. We would expect that the d exact roots of $p(x)$ would be reasonably close to d of the n roots of $\hat{p}(x)$, but where will the $n-d$ extra roots be? A perturbation analysis as $\|e\| \to 0$ is illuminating.

Theorem 4.2. *The $n-d$ extra roots grow like $1/\|e\|^{1/(n-d)}$ as $\|e\| \to 0$. Moreover, the roots are asymptotically symmetric, in the sense that they approach infinity along the $n-d$ rays of a star with equal angles and at equal radii.*

Proof. Without loss of generality, consider the family of polynomials

$$p(x) = c \prod_{j=1}^{k} \left(\frac{x}{R} - r_j \right) \prod_{j=k+1}^{n} (x - r_j) \,.$$

This is without loss of generality because any given polynomial may be embedded in such a family of polynomials by appropriate choice of the $O(1)$ constants r_k. Expanding the product of the first k factors gives (with $\xi = x/R$)

$$\xi^k - (r_1 + r_2 + \cdots + r_k)\xi^{k-1} + \cdots + (-1)^k r_1 r_2 \cdots r_k \,.$$

If it so happens that $r_1 + r_2 + \cdots + r_k = 0$ and likewise for all the (symmetric) sums of the intermediate terms except the last, we are left with a factor

$$\left(\frac{x}{R}\right)^k + (-1)^k r_1 r_2 r_3 \cdots r_k \,.$$

This differs from $O(1)$ by a term $O(1/R^k)$. Therefore $p(x_j) = p_{n-k}(x_j)(1 + O(1/R^k))$, making only a small change in the values of p at the nodes x_j. Moreover, the only way that we can have all the intermediate symmetric sums zero is if r_1, r_2, …, r_k are kth roots of a constant, which establishes the second part of the theorem. $\square$

Remark. The computation shows that we may take any degree $d < n$ polynomial, evaluate it at $n+1$ nodes, perturb the values at those nodes by $O(1/R^{n-d})$, and introduce zeros lying near a circle of radius R. This is typically seen by having rounding error perturbations of size μ, the machine epsilon (typically something like 2^{-53} or 10^{-16}), and seeing spurious roots near a circle of radius $\mu^{-1/k}$, where k is the number of excess nodes. For example, if $n = 15$, and the nodes are the 16th roots of unity, and the polynomial being sampled is 1 (so $d = 0$ and all roots are spurious), the computed eigenvalues all have norm about 8.3, and are symmetrically placed about the origin. For comparison, $(2^{-46})^{1/15} \approx 8.3$, and separate investigation of Maple's machine epsilon reveals that on this machine, we have $\mu \approx 2^{-46}$.

4.4. Pseudospectra

The thesis [2] contains the following lemma, which extends a similar one for the monomial basis [23] to the case of arbitrary bases. We can therefore separate the spectral properties inherent to the matrix polynomial $\mathbf{P}(x)$ from those arising from the choice of representation. For an example, see Fig. 2.

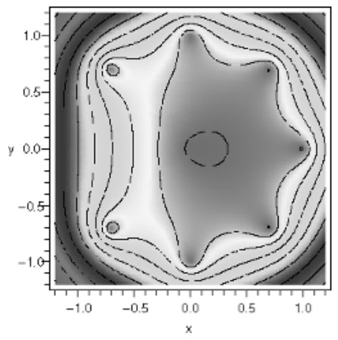


FIGURE 2. Nodes at origin and at the eighth roots of unity, *omitting* -1, have a benign influence on pseudospectra where $B(z) = \sum_{k=0}^{n} \alpha_k |\phi_k(z)|$ is small. In the centre region here, $B \approx 1$, while for each four (logarithmic) contour lines, $B(z)$ grows by a factor of about 10.

The *weighted pseudospectrum* of $\mathbf{P}(x) = \sum_{k=0}^{n} \mathbf{C}_k \phi_k(x)$ is defined to be

$$\Lambda_\varepsilon(\mathbf{P}) := \{z \in \mathbb{C} : \det((\mathbf{P} + \mathbf{\Delta P})(z) = 0 \text{ with } \|\Delta \mathbf{C}_k\| \le \alpha_k \varepsilon\} \ ,$$

where the weights α_k are nonnegative and not all zero. The lemma says that this set can also be described as

$$\Lambda_\varepsilon(\mathbf{P}) = \left\{z \in \mathbb{C} : \|\mathbf{P}^{-1}(z)\| \ge (\varepsilon B(z))^{-1}\right\} \ ,$$

where the scalar function $B(z) := \sum_{k=0}^{n} \alpha_k |\phi_k(z)|$ depends on the weights and on the (scalar) basis functions $\phi_k(z)$, but not on $\mathbf{P}$. This formula contains quantities that also appear in the condition number of Sect. 2.5.

Pseudospectral results will be explored further in a future paper. See, meanwhile, the poster [4].

4.5. Lebesgue Constants

This paper has concentrated on complex nodes, especially near-circular nodes (because of their beautiful conditioning, and their natural character of surrounding points of interest). However, even more work has been done on sets of real nodes, for obvious reasons. We summarize some of the implications of these standard results here.

By abstracting out the values of the polynomials being interpolated, we arrive at a generic characterization of interpolation nodes by themselves: the Lebesgue constants.

$$\Lambda_\Omega(x_0, x_1, \ldots, x_n) := \sup_{x \in \Omega} \sum_{k=0}^{n} \frac{|\prod_{j \ne k}(x - x_j)|}{|\prod_{j \ne k}(x_k - x_j)|} \tag{4.1}$$

This is a measure of how good or bad the Lagrange polynomials based on the nodes $\{x_k\}$ are at representing arbitrary polynomials in the region Ω.

Remarks.

1. The Lebesgue constants are the maximum possible condition number for a polynomial taking on values with $|p_k| = 1$.
2. It is possible that some functions can be well represented for rootfinding or for evaluation, by interpolation at the x_k in Ω, even if the Lebesgue constant is large for those nodes in Ω; the Lebesgue constants represent the worst possible behaviour.
3. A lower bound for the Lebesgue constants on $n+1$ nodes is given in [6]:
$$\Lambda \geq \frac{1}{2n^2}\frac{\max w_k}{\min w_k} \,.$$
This shows that if the barycentric weights w_k vary widely, then the interpolation problem is ill-conditioned on these nodes.

5. Concluding Remarks

One interesting problem in Polynomial Algebra By Values is "what is the degree"? If the degree is known *a priori* then many algorithms (particularly that of division) can be carried out without difficulty. However, what if the degree is not known? How does one deduce the degree, *without converting to a monomial basis*? This last restriction disallows least-squares fitting, for example. One possible suggestion is to use the perturbation result of Sect. 4.3: compute the roots (!) and count the ones that are not large enough to be perturbations of a multiple root at infinity. Another is to use dual norms to find the *nearest polynomial of lower degree*, much as one finds the nearest polynomial with a given root [25]. This approach will be explored in a future paper.

Another interesting problem is that further work is needed to adapt the proofs of [12, 29] to show that standard numerical methods do, in fact, give the exact roots of a polynomial (polynomial eigenvalues of a matrix polynomial) nearby in the Lagrange basis. Since there are twice as many degrees of freedom as in the monomial case (here, we may also perturb the nodes), one expects this to be possible.

There are a surprising number of elementary questions remaining to be addressed, now that it seems reasonable to work with polynomials entirely by values.

References

[1] A. Amiraslani. Dividing polynomials when you only know their values. In Laureano Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 5–10, June 2004.

[2] A. Amiraslani. *Algorithms for Matrices, Polynomials, and Matrix Polynomials.* PhD thesis, University of Western Ontario, London, Canada, May 2006.

[3] A. Amiraslani, D. Aruliah, and R.M. Corless. The Rayleigh quotient iteration for generalized companion matrix pencils. *in preparation*, 2006.

[4] A. Amiraslani, D. Aruliah, R.M. Corless, and N. Rezvani. Pseudospectra of matrix polynomials in different bases. Poster TR-06-03, Ontario Research Centre for Computer Algebra, http:// www.orcca.on.ca /techreports, June 2006. presented as a poster at CAIMS/MITACS York, June 2006.

[5] A. Amiraslani, R.M. Corless, L. Gonzalez-Vega, and A. Shakoori. Polynomial algebra by values. Technical Report TR-04-01, Ontario Research Centre for Computer Algebra, http:// www.orcca.on.ca/ TechReports, January 2004.

[6] J.-P. Berrut and L.N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3): 501–517, 2004.

[7] D.A. Bini and L. Gemignani. Bernstein-Bezoutian matrices. *Theor. Comput. Sci.*, 315(2-3): 319–333, 2004.

[8] D.A. Bini, L. Gemignani, and V.Y. Pan. Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations. *Numer. Math.*, 100: 373–408, 2005.

[9] R.M. Corless. Generalized companion matrices in the Lagrange basis. In Laureano Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 317–322, June 2004.

[10] R.M. Corless. The reducing subspace at infinity for the generalized companion matrix in the Lagrange basis. *in preparation*, 2006.

[11] R.M. Corless and S.M. Watt. Bernstein bases are optimal, but, sometimes, Lagrange bases are better. In *Proceedings SYNASC, Timisoara*, pages 141–153. MITRON Press, September 2004.

[12] A. Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210): 763–776, April 1995.

[13] R.T. Farouki and T.N.T. Goodman. On the optimal stability of the Bernstein basis. *Math. Comput.*, 65(216): 1553–1566, 1996.

[14] R.T. Farouki and V.T. Rajan. On the numerical condition of polynomials in Bernstein form. *Comput. Aided Geom. Des.*, 4(3): 191–216, 1987.

[15] R.T. Farouki and V.T. Rajan. Algorithms for polynomials in Bernstein form. *Comput. Aided Geom. Des.*, 5(1): 1–26, 1988.

[16] S. Fortune. Polynomial root finding using iterated eigenvalue computation. In Bernard Mourrain, editor, *Proceedings ISSAC*, pages 121–128, London, Canada, 2001. ACM Press.

[17] T. Hermann. On the stability of polynomial transformations between Taylor, Bézier, and Hermite forms. *Numerical Algorithms*, 13: 307–320, 1996.

[18] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[19] N.J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA Journal of Numerical Analysis*, 24: 547–556, 2004.

[20] T.E. Hull and R. Mathon. The mathematical basis and a prototype implementation of a new polynomial rootfinder with quadratic convergence. *ACM Trans. Math. Softw.*, 22(3): 261–280, 1996.

[21] G.F. Jónsson and S. Vavasis. Solving polynomials with small leading coefficients. *Siam Journal on Matrix Analysis and Applications*, 26(2): 400–414, 2005.

[22] T. Lyche and J.M. Peña. Optimally stable multivariate bases. *Advances in Computational Mathematics*, 20: 149–159, January 2004.

[23] F. Tisseur and N.J. Higham. Structured pseudospectra for polynomial eigenvalue problems, with applications. *SIAM J. Matrix Anal. Appl.*, 23(1): 187–208, 2001.

[24] V.Y. Pan. Coefficient-free adaptations of polynomial root-finders. *Computers and Mathematics with Applications*, 50: 263–369, 2005.

[25] N. Rezvani and R.M. Corless. The nearest polynomial with a given zero, revisited. SIGSAM BULLETIN*, Communications on Computer Algebra*, 134(3): 71–76, September 2005.

[26] O. Ruatta. A multivariate Weierstrass iterative rootfinder. In *ISSAC '01: Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pages 276–283, New York, NY, USA, 2001. ACM Press.

[27] A. Shakoori. The Bézout matrix in the Lagrange basis. In Laureano Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 295–299, June 2004.

[28] B.T. Smith. Error bounds for zeros of a polynomial based upon Gerschgorin's theorem. *Journal of the Association for Computing Machinery*, 17(4): 661–674, October 1970.

[29] K.-C. Toh and L.N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numerische Mathematik*, 68: 403–425, 1994.

[30] Y.-F. Tsai and R.T. Farouki. Algorithm 812: BPOLY: An object-oriented library of numerical algorithms for polynomials in Bernstein form. *ACM Trans. Math. Softw.*, 27(2): 267–296, 2001.

[31] J.R. Winkler. A comparison of the average case numerical condition of the power and Bernstein polynomial bases. *Intern. J. Computer Math.*, 77: 583–602, 2001.

[32] J.R. Winkler. The transformation of the companion matrix resultant between the power and Bernstein polynomial bases. *Appl. Numer. Math.*, 48(1): 113–126, 2004.

Acknowledgment

Many people have helped with this work: I particularly acknowledge Dhavide Aruliah and Laureano Gonzalez-Vega, and my students Amir Amiraslani, Nargol Rezvani, and Azar Shakoori. I also thank Dario Bini, Martin Kreuzer, Michael Möller, Lorenzo Robbiano, Tomas Sauer, and Joab Winkler, for their interest in this approach and their encouragement.

Robert M. Corless
Distinguished University Professor
Ontario Research Centre for Computer Algebra
and the Department of Applied Mathematics
University of Western Ontario
1151 Richmond St.
London, Canada
e-mail: `rcorless@uwo.ca`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 17–45
© 2007 Birkhäuser Verlag Basel/Switzerland

Ill-conditioned Properties and Hybrid Computations

Matu-Tarow Noda

Abstract. Approximate algebraic computation (AAC) has been one of the most important research areas in algebraic computation. The basis of AAC is an algorithm of computing approximate greatest common divisors (AppGCD) proposed by T. Sasaki and the author. AppGCD and its applications work well, especially, for obtaining accurate results of ill-conditioned problems. Algorithms and implementation methods of AppGCD are briefly surveyed and its applications such as hybrid integral, hybrid rational function approximation (HRFA), data smoothing by using HRFA and new hybrid method for computing Cauchy principal value integral are described. Further, a pathological feature of HRFA and relations of HRFA and ill-conditioned problems, and their applications are discussed.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 33F10.

Keywords. Hybrid computation, ill-conditioned property, algebraic equation, rational interpolation, quadrature.

1. Introduction

Traditionally, the word **scientific computation** means computation done numerically, i.e., numerical computation. The successes of numerical computation have brought about today's development of the *Computer World*. Numerical computations have a very wide application area, and a number of research works on applied mathematics have also been proposed. Further, this research has supported current developments of the so-called engineering society. However, as is well known, these developments have some defects. They include:

1. the results are always in danger of numeric error,
2. since a number of algorithms are proposed, problems related to algorithm selection occur, and
3. algebraic or symbolic computations are impossible.

On the other hand, rapid progress of computer hardware and software has led to the use of symbolic computation. Symbolic computation wastes a significant amount of computer memory and computing time as well. Computations done symbolically give exact results. Thus, if numerical computation is effectively combined with symbolic computation, some problems in numerical computation may be solved more quickly and accurately.

Many approaches have been used for combining numerical computation with symbolic computation. Some of them are shown by Kaltofen *et al.* [13, 4]. One of the candidates for such type of computation that may be used effectively is the problem of solving ill-conditioned polynomial equations. A polynomial is said to be ill-conditioned if small changes in its coefficients result in large changes in its zeros. An ill-conditioned polynomial equation has at least one of the following properties:

1. the existence of several roots having ratios close to unity,
2. the existence of multiple roots.

The first property means the existence of close zeros in a polynomial equation. For multiple roots, let the polynomial equation be

$$P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = 0\ , \quad a_n \neq 0,$$

and let m_j be the multiplicity of a solution x_j of $P_n(x) = 0$. If a coefficient a_k is perturbed slightly to $a_k + \Delta a_k$, then the perturbation in x_j is

$$x_j = \left\{ -\frac{m_j! x_j^{n-k} \Delta a_k}{(d/dx)^{m_j} P_n(x_j)} \right\}^{1/m_j} ,$$

where $\Delta a_k \ll a_k$ [2]. Many numerical algorithms have been proposed to obtain zeros effectively. However, most of them are not effective for ill-conditioned problems.

If the polynomial $P_n(x)$ has integer coefficients and its roots are integer, then multiple roots are easily separated by symbolic computation. For s-fold multiple roots, $P_n(x)$ is divided by $d^{(s)}P_n(x)/dx^s$, with residual equal to zero. It follows that the greatest common divisor (GCD) of $P_n(x)$ and its $(s-1)$-times differentiation by x are not primitive. The GCD of two univariate polynomials P_1 and P_2 with exact coefficients, $\mathrm{GCD}(P_1, P_2)$, is obtained by algebraic Euclidean algorithm. Sasaki and the author applied the Euclidean algorithm to polynomials whose coefficients are inexact, i.e., with limited accuracy or perturbed within small tolerance. In this case, GCD is replaced by approximate GCD (AppGCD) with accuracy ε, $\mathrm{AppGCD}(P_1, P_2; \varepsilon)$ [26, 21]. There are two simple approaches to computing AppGCD for polynomials with inexact coefficients. One is the method known as AppGCD (with accuracy ε) and the other is the method using interval arithmetic [20]. These two methods will be briefly described in Sect. 2. AppGCD has been applied to several kinds of scientific computations. The first is an integral of a given ill-behaved function which is obtained in the symbolic-numeric combined environment. AppGCD plays an important role in the integration procedure. The method is called *hybrid integral* and will be described in Sect. 3. Section 4 is devoted to hybrid rational function approximation, which is abbreviated simply as HRFA. In HRFA, AppGCD is effectively used. If a given function or set of data is approximated by a rational function, there often occur approximate common factors in numerator and denominator polynomials of the rational function. These approximate common factors are eliminated by the AppGCD of the numerator and denominator polynomials. Hybrid algorithms are applied to some practical problems such as data smoothing and Cauchy principal value integral. These applications will be shown in Sect. 5.

2. Approximate GCD Computations

Two implementation methods for AppGCD are discussed. One is known as AppGCD with accuracy ε, which is the basis of every kind of AppGCD proposal. The other is AppGCD using interval arithmetic. After the successful application of AppGCD for solving an ill-conditioned algebraic equation, other AppGCDs have been proposed by several researchers from different viewpoints. They are also summarized briefly.

2.1. Approximate GCD with Accuracy ε

The AppGCD of two polynomials P_1 and P_2 with accuracy ε, AppGCD$(P_1, P_2; \varepsilon)$, is a natural extension of the usual GCD computation by Euclidean algorithm. Here, coefficients of both polynomials P_1 and P_2 are inexact and are represented by floating point numbers. The polynomial remainder sequence (PRS) is obtained by the Euclidean algorithm as follows:

$$P_{i-1} = P_i Q_i + P_{i+1}, \qquad i = 2, \dots,$$

where Q_i is a quotient polynomial. Because coefficients are inexact, all coefficients of the polynomials in the PRS contain error by division operations. A cutoff operation which regards the coefficients of polynomials in the PRS smaller than a tolerance ε as zero is introduced. Thus the PRS is written as

$$P_1, P_2, P_3, \dots, P_k \neq 0, P_{k+1} = 0 \text{ (cutoff } \varepsilon).$$

The approximate GCD with accuracy ε is defined as

$$\mathrm{AppGCD}(P_1, P_2; \varepsilon) = P_k.$$

The above procedures are shown in an algorithm below:

Algorithm 1: Approximate GCD Algorithm

Input: Univariate regular polynomials $P_1(x)$ and $P_2(x)$ with $\deg(P_1) > \deg(P_2)$ and a small positive number ε.

Output: Approximate GCD of $P_1(x)$ and $P_2(x)$ with accuracy less than ε, AppGCD(P_1, P_2, ε).

Algorithm:

1. Calculate a PRS

$$P_1,\ P_2, \ldots, P_k \neq 0 \text{ (cutoff } \varepsilon),\ P_{k+1} = 0 \text{ (cutoff } \varepsilon)$$

   by the iteration formula

$$\begin{aligned} Q_i &= \text{quotient}(P_{i-1}, P_i)\,; \\ P_{i-1} &= Q_i P_i + \max\{1, \text{mmc}(Q_i)\} \times P_{i+1}, \quad i = 2, \ldots, k\,, \end{aligned}$$

   where mmc denotes the maximum magnitude coefficient of the polynomial.
2. Return the primitive part (pp) of P_k as

$$\text{AppGCD}(P_1(x), P_2(x); \varepsilon) \ = \ \text{pp}(P_k).$$

The univariate AppGCD has been used to solve ill-conditioned polynomial equations (as shown in detail in [26]) and extended to obtain the AppGCD of multivariate polynomials [22, 21].

2.2. Approximate GCD by Interval Arithmetic

The PRS is computed by using the circular interval arithmetic. An inexact floating-point number is considered to be a pair of its center, M, and an error radius, $r(M)$, which corresponds to a perturbation with small tolerance in a circular interval number. Thus the circular interval number is represented as $< M, r(M) >$. The circular interval arithmetic is defined as arithmetic between circular interval numbers. The PRS computation is similar to that of AppGCD with accuracy ε. The difference is on a stopping criterion of the PRS. In AppGCD with ε, the cutoff operation is introduced to regard small coefficients as zero. However, in the interval arithmetic, the condition that the circle contains zero is used as the criterion. There arises a difficulty on division by an interval number. The fact that the denominator must not include zero in the interval arithmetic. In the circular interval arithmetic case, if the condition

$$\mid M \mid < \mid r(M) \mid$$

shows the circular interval contains zero. Thus the PRS is stopped at a polynomial whose coefficient satisfies the above condition. Later, Shirayanagi and Sweedler discussed the condition in detail with rigorous mathematical proofs by using the rectangular interval arithmetic and established a theory to stabilize algebraic algorithms [30].

2.3. Computation Examples of Both AppGCDs

Consider the polynomial equation

$$P(x) = x^4 - 10.4x^3 - 70.96x^2 + 29.6x - 3 = 0.$$

It has roots $x = -5, 15$ and a double root at $x = 0.2$. The polynomial equation is ill-conditioned because of its double root. The two methods mentioned above may be applied to solve the equation. The Eucledian algorithm with inexact coefficients gives the following PRS:

$$\begin{aligned}
P_1 &= x^4 - 10.4x^3 - 70.96x^2 + 29.6x - 3,\\
P_2 &= 4x^3 - 31.2x^2 - 141.92x + 29.6 \quad (= dP_1/dx),\\
P_3 &= -85.78x^2 - 107.76614385x + 24.98461538,\\
P_4 &= -0.46563934x + 0.09312787,\\
P_5 &= 2.77555756 \times 10^{-17}.
\end{aligned}$$

The maximal value of P_5 is less than 10^{-16} times than that of P_4. Thus, the cutoff operation regards P_5 as zero. After a normalization of the head term of P_4, the AppGCD with accuracy $\varepsilon = 10^{-16}$ is obtained and shown as

$$\mathrm{AppGCD}(P_1, P_2; 10^{-16}) = x - 0.2.$$

On the other hand, the PRS generated by the interval arithmetic shows

$$\begin{aligned}
P_1 =& < 1.0, 2.2 \times 10^{-16} > x^4 + < -10.4, 1.8 \times 10^{-15} > x^3\\
&+ < -70.96, 1.4 \times 10^{-14} > x^2 + < 29.6, 3.6 \times 10^{-15} > x + < -3.0, 4.4 \times 10^{-16} >,\\
P_2 =& < 4.0, 1.8 \times 10^{-15} > x^3 + < -31.2, 8.9 \times 10^{-15} > x^2\\
&+ < -141.92, 5.7 \times 10^{-14} > x + < 29.6, 7.1 \times 10^{-15} >,\\
P_3 =& < -8.9 \times 10^2, 3.8 \times 10^{-12} > x^2 + < -1.1 \times 10^3, 9.3 \times 10^{-12} > x\\
&+ < 2.6 \times 10^2, 1.8 \times 10^{-12} >,\\
P_4 =& < -4.7 \times 10^6, 1.1 \times 10^{-7} > x + < 9.5 \times 10^5, 2.2 \times 10^{-8} >,\\
P_5 =& < 1.3 \times 10^{-6}, 7.8 \times 10^{-4} > \ni 0.
\end{aligned}$$

PRS computations terminate because P_5 contains zero in its interval. By the algorithm stabilization technique [30], P_5 should be rewritten to zero (zero rewriting). Then, we obtain

$$\mathrm{GCD}(P_1, P_2) = \mathrm{GCD}(P(x), dP(x)/dx) = x - < 0.2, 4.7 \times 10^{-10} > .$$

Both GCDs give approximate double root of $P(x)$ that exists very close to $x = 0.2$. Many kinds of algorithms to obtain approximate GCDs have been proposed. It follows that approximate GCD and its applications become one of the most interesting research subjects of computer algebra.

2.4. Solving Ill-conditioned Algebraic Equation

The first success of applications of AppGCD is to solve a univariate ill-conditioned algebraic equation. A polynomial equation, $P(x) = 0$, is called *ill-conditioned* when the equation has multiple and/or close roots as mentioned in Sect. 1. If the coefficients of $P(x)$ are integers or rational numbers, say *exact*, multiple roots are easily found and separated by the traditional GCD operation based on the Euclidean algorithm. However, if coefficients are *inexact* and represented as floating-point numbers, it becomes impossible to use traditional GCD and then, the equation should be solved numerically. But, in numerical computation, close roots reduce the accuracy of all roots of the equation. Then, we should make an algorithm which separates not only multiple but also close roots effectively for the algebraic equation with *inexact* coefficients. This was the first motivation to propose the AppGCD algorithm [26].

The given polynomial $P(x)$ is, first, decomposed to a square-free form by using AppGCD with accuracy ε. The square-free decomposition of $P(x)$ is written as

$$P(x) = Q_1(x)Q_2^2(x)\cdots Q_m^m(x), \tag{2.1}$$

where $Q_m^m(x)$ contains all the m-multiple factors of $P(x)$; hence each Q_i has no multiple factor. To obtain (2.1), the following method is used:

$$\mathrm{AppGCD}\left(P(x), \frac{dP(x)}{dx}, \varepsilon\right) = Q_2(x)Q_3^2(x)\cdots Q_m^{m-1}(x). \tag{2.2}$$

Dividing (2.1) by (2.2), the product of square-free factors are obtained as

$$Q_1(x)Q_2(x)\cdots Q_m(x).$$

With repeated use of the above procedure, $P(x)$ can be separated to $Q_1, Q_2, \ldots, Q_l$. The procedure by using AppGCD may be called *approximate square-free decomposition.*

After the square-free decomposition, for each $Q_l(x),\ l = 1, \ldots, m,\ Q_l(x) = 0$ is solved by rough, numerical computation. Let the roots of the computation be $u_{l_1}, u_{l_2}, \ldots$. For each u in $u_{l_1}, u_{l_2}, \ldots$, expand $P(u+v)$ up to v^l terms by Taylor expansion as

$$P(u) + P^u\frac{v}{1!} + \cdots + P^l(u)\frac{v^l}{l!} + O(v^{l+1}) = 0, \tag{2.3}$$

where

$$P^{(l)}(u) = \frac{d^lP(x)}{dx^l}\,|_{x=u}\ .$$

The expansion (2.3) gives an equation on v, $p(v) = 0$. Since the degree of the equation is l, l solutions, $v_1, \ldots, v_l$, may be obtained. By adding these l solutions to the solution of $Q_l(x) = 0$, all roots of $P(x) = 0$ are obtained with high accuracy. We show an example of how to solve the univariate ill-conditioned algebraic equation by the method mentioned above [21]. Let an algebraic equation be

$$\begin{aligned} P(x) &= x^7 - 3.504x^6 + 0.762003x^5 + 6.87799x^4 - 4.02601x^3 \\ &\qquad - 2.62198x^2 + 2.51201x - 0.504006 \\ &= (x+1)^2(x-2)^2(x-0.5)(x-0.501)(x-0.503) = 0. \end{aligned} \tag{2.4}$$

The equation (2.4) has two double roots at $x = -1$ and $x = 2$, and also three close roots around $x \simeq 0.5$. Our method is divided into three steps.

1. Obtain the AppGCD with accuracy ε
PRS is obtained as

$$
\begin{aligned}
P_1 &= P(x),\\
P_2 &= \frac{dP(x)}{dx} = 7x^6 - 21.024x^5 + 3.81001x^4 + 27.512x^3 - 12.078x^2 - 5.24397x + 2.51201,\\
P_3 &= -1.28572x^5 + 3.22017x^4 - 0.333188x^3 - 2.73655x^2 + 1.77815x - 0.324372,\\
P_4 &= 1.23979x^4 - 2.48289x^3 - 0.924883x^2 + 2.17459x - 0.623204,\\
P_5 &= -3.37499 \times 10^{-6}x^3 + 5.06537 \times 10^{-6}x^2 + 5.505959 \times 10^{-6} - 3.38077 \times 10^{6}\\
&\simeq 0 \text{ cutoff } 10^{-4}.
\end{aligned}
$$

Thus the AppGCD with $\varepsilon = 10^{-4}$ is obtained as

$$
\begin{aligned}
\mathrm{AppGCD}(P, P', 10^{-4}) &= \mathrm{pp}(P_4)\\
&= x^4 - 2.0026667x^3 - 0.74599899x^2 + 1.7539990x - 0.50266867,
\end{aligned}
$$

where P' denotes $dP(x)/dx$.

2. Approximate square-free decomposition
By using the AppGCD obtained above, the following square-free factors are computed:

$$Q_3(x) = x - 0.50133334, \quad Q_2(x) = x^2 - x - 2.0.$$

Then $P(x)$ is decomposed as

$$P(x) = Q_2^2(x)Q_3^3(x) = (x^2 - x - 2.0)^2(x - 0.50133334)^3. \tag{2.5}$$

Roots of Q_2 are easily obtained as $x = u_{2_1} = -1$ and $x = u_{2_2} = 2$. On the other hand, a common factor Q_3 gives $x = u_3 = 0.50133334$. These solutions may correspond to two double roots and close roots of $P(x) = 0$.

3. Expand procedure of close roots
The remaining problem is to obtain the detailed behavior of $P(x)$ at $x \simeq -1, 2$ and 0.50133334. The first, a root $x = -1$, is considered. $P(x + v)$ is expanded up to v^2 at $x = -1$ and one obtains the result

$$v^2 - 1.16651 \times 10^{-16}v + \text{small term } (\ll 10^{-16}) = 0.$$

Thus, $v = 0$ is obtained with a sufficient accuracy. Thus, the root $x = -1$ is a double root of $P(x) = 0$. Similar procedure may be applied to the root $x = 2$. Then, the root $x = 2$ is also a double root of $P(x) = 0$. The next, the approximate triple factor $Q_3(x)$, is considered. Taylor expansion of $P(x + v)$ at $x = u_3$ up to v^3 gives

$$5.0625v^3 + 1.32583 \times 10^{-7}v^2 - 1.18125 \times 10^{-5} - 3.75007 \times 10^{-9} = 0.$$

Roots of this equation are

$$v_1 = -0.00133334, \quad v_2 = -0.00033334 \quad \text{and} \quad v_3 = 0.001666666.$$

By adding these values to u_3, the following results are obtained:

$$x = 0.5, \quad x = 0.501 \quad \text{and} \quad x = 0.503.$$

They are the three close roots of $P(x) = 0$ with high accuracy.

The method above can be straightly applied to solve a system of multivariate algebraic equations. In this case, coefficients of equations are also *inexact* and the system is ill-conditioned. Detailed discussions are described in the papers [22, 21].

2.5. Other Approximate GCDs

Before the proposal of the paper by Sasaki and the author [26], similar approaches have been discussed by Schönhage [28], Auzinger and Stetter[1]. However, the approach in the former study is considerably different from ours, and the discussions in the paper are devoted mostly to the computation time complexity of the algorithm. Further, coefficients of input polynomials are assumed to be arbitrarily precise, i.e., belonging to *exact*. The authors of the latter paper describe a method of solving a system of algebraic equations by using a numerical resultant algorithm.

Thus, in this paper, six different approaches for the approximate GCD are briefly reviewed.

Approximate GCD proposed by Sederberg and Chang [29]**:** The algorithm is considered and applied as a numeric-symbolic algorithm to a problem of computer-aided design. Let $P_1(x), P_2(x), \ldots, P_n(x)$ be polynomials. The set of polynomials is perturbed so as to induce a linear common factor. That is, for a set of the perturbation polynomials $\varepsilon(x) = \varepsilon_1(x), \ldots, \varepsilon_n(x)$, $\gcd(P_1(x)+\varepsilon_1(x), P_2(x)+\varepsilon_2(x), \ldots, P_n(x)+\varepsilon_n(x))$ induce a linear polynomial, common factor. A norm of perturbation polynomial is written as

$$\|\varepsilon\|_{[a,b]} = \max_{a\le x\le b}\sqrt{\sum_{i=1}^{n}\varepsilon_i^2(x)}$$

over a prescribed parameter interval $[a, b]$. If we represent $\varepsilon_i(x)$ by the Chebyshev polynomials, we can determine ε which gives the minimum norm, uniquely. The result is applied to the problem of approximating high degree curves by small degree ones.

Approximate GCD proposed by Corless, Gianni, Trager and Watt [3]**:** For two given univariate polynomials $P_1(x)$ and $P_2(x)$, write the Sylvester matrix of the coefficients of $P_1(x)$ and $P_2(x)$. The Singular Value Decomposition (SVD) technique is applied to the matrix to compute an upper bound on the degree of the approximate GCD. By SVD, if the matrix becomes rank deficient by one, then there exists a common factor of both polynomials. Further, the number of rank deficient rows shows the degree of the GCD. The method is applied to a multivariate polynomial case. In the paper [3], an optimization problem is first used on the problem of obtaining approximate GCD. The Euclidean norm (2-norm) is used and this selection of the norm is in the paper.

Approximate GCD proposed by Karmarkar and Lakshman [14]**:** The approach to find the approximate GCD is similar to the above-mentioned method proposed by Corless *et al.* [3]. Two monic polynomials $P_1(x), P_2(x) \in C[x]$ with $\deg(P_1) = m, \deg(P_2) = n$, where $C[x]$ denotes a polynomial with complex coefficients, are considered. The problem is stated as to find polynomials $\hat{P}_1, \hat{P}_2 \in C[x]$ with $\deg(\hat{P}_1) < m, \deg(\hat{P}_2) < n$ such that $P_1 + \hat{P}_1$ and $P_2 + \hat{P}_2$ have a non-trivial GCD, and $\|\hat{P}_1\| + \|\hat{P}_2\|$ is minimized. The norm used here is also the Euclidean norm. The running time of the algorithm is a polynomial of the degrees m, n. Further, a concept of the nearest singular polynomial to P_1, that is a polynomial $h \in C[x]$ with $\deg(h) = m$ such that h has a double root and $\|\hat{P}_1\|$ is minimized, where $\hat{P}_1 = h - P_1$, is proposed. Many works on the nearest singular polynomial have been considered.
Approximate GCD proposed by Hribernig and Stetter [7]**:** The approximate GCD is here called the *near-GCD*. For two given polynomials $P_1, P_2 \in C[x]$, at the accuracy level α, the polynomials possesss a near-GCD $\tilde{g}$ if there exist polynomials $P_1^*, P_2^* \in C[x]$ satisfying

$$\mathrm{GCD}(P_1^*, P_2^*) = \tilde{g}, \quad \text{and} \quad \|\tilde{P}_i - P_i^*\| \leq \alpha, \quad i = 1, 2.$$

Equivalently, a near-GCD $\tilde{g}$ of $\tilde{P}_1$ and $\tilde{P}_2$ at accuracy level α is denoted as α-GCD$(\tilde{P}_1, \tilde{P}_2)$ and computed via the Euclidean algorithm. Here, as the norm of the polynomial, the 1-norm is used. The algorithm gives a lower bound on the degree of the approximate GCD.
Approximate GCD proposed by Emiris, Galligo and Lombardi [5]**:** Several Approximate GCDs of two univariate polynomials obtained by the above algorithms depend on the rounding mode or the accuracy ε selected for floating-point computations. Further, as mentioned above, algorithms based on Euclidean algorithm give the lower bound of the degree of the approximate GCD. The purpose of the algorithm, developed here, is to obtain the maximum-degree approximate GCD depending on the given tolerance ε. For two given polynomials $P_1, P_2 \in C[x]$, whose degree is n and m $(n \geq m)$ respectively, and torelance $\varepsilon \in (0, 1]$, upper bound on the degree of the approximate GCD, ε-GCD of P_1, P_2, are obtained. The degree of the ε-GCD is defined to be the maximum integer r such that there exist $\hat{P}_1, \hat{P}_2 \in C[x]$ of degree bounded by n and m, respectively, with $\mid P_1 - \hat{P}_1 \mid, \mid P_2 - \hat{(P_2)} \mid \leq \varepsilon$ and deg(GCD$(\hat{P}_1, \hat{P}_2)$)$= r$. To guarantee the maximum-degree approximate GCD, SVD computations on subresultant matrices and a gap theorem are effectively used. The norm used here is the Euclidean norm (2-norm).
Approximate GCD proposed by Pan [23]**:** Several methods for computing the approximate GCD have been proposed by Pan. Here, for $P_1(x)$ and $P_2(x)$, polynomials P_1^* and P_2^* are considered. The approximate GCD of P_1^* and P_2^*, GCD(P_1^*, P_2^*), satisfies the following relations for a real value b:

$$\deg(P_1^*) \leq \deg(P_1(x)), \qquad \deg(P_2^*) \leq \deg(P_2(x)), \tag{2.6}$$

$$\|P_1^* - P_1(x)\| \leq 2^{-b}\|P_1(x) \qquad \|, \|P_2^* - P_2(x)\| \leq 2^{-b}\|P_2(x)\|.$$

The maximum δ-GCD satisfies the relations in (2.6). Let the two given polynomials be

$$P_1(x) = u \times \prod_{i=1}^{n}(x - y_i), \quad \text{and} \quad P_2(x) = v \times \prod_{j=1}^{m}(x - z_j),$$

where u and v are the leading coefficients of $P_1(x)$ and $P_2(x)$, respectively. Thus, y_i and z_j are zeros of $P_1(x)$ and $P_2(x)$. Further, a small constant δ is given. The maximum δ-GCD $\tilde{g}_\delta(x)$ of $P_1(x)$ and $P_2(x)$ is defined as

$$\tilde{g}_\delta(x) = \prod_{k=1}^{r}(x - x_k), \qquad x_k = \frac{y_{i_k} + z_{j_k}}{2}, \quad k = 1, \ldots, r,$$

where the set of pairs $(y_{i_1}, z_{j_1}), \ldots, (y_{i_r}, z_{j_r})$ is a maximum matching which maximizes r and satisfies $\mid y_{i_k} - z_{j_k} \mid \leq 2\delta$. If δ is bounded by $\delta \leq (1 + 2^{-b})^{1/n} - 1$, then the polynomials $P_1^* = P_1(x)\tilde{g}_\delta(x)$ and $P_2^* = P_2(x)\tilde{g}_\delta(x)$ satisfy (2.6). The maximal norm is used to define the maximum δ-GCD.

After successful introduction of approximate GCDs as described above, many algorithms for computing approximate GCDs have been proposed. There are also new developments on approximate algebraic computation such as computing nearest singular polynomials and approximate factorization. Most of the algorithms are based on AppGCD. They have been published in recent proceedings of international conferences on computer algebra and are summarized in [13, 4].

3. Approximate GCD and Hybrid Integral

Obtaining an indefinite integral of a given function is one of the most important operations in scientific computation. If the function is given by a rational function with exact coefficients, it can be integrated symbolically. There remains, however, several cases such as (1) even if the function is given, a closed form solution is not obtainable, (2) the function is defined by a table or a set of discrete data, and (3) the function has inexact floating-point coefficients with limited accuracy. Numerical algorithms give only numerical results for definite integrals. Here we show an algorithm of hybrid integral in which numerical methods are effectively combined with an algorithm of symbolic (algebraic) integral. The hybrid integral algorithm gives symbolic results, a kind of *indefinite integral*, for given functions. Accurate numerical results of definite integrals are easily obtained by simple substitutions of upper and lower bounds of integrals into symbolic results. The algorithm proposed here is an extension of an algebraic algorithm of integration for rational function by Horowitz [6] to the *inexact* coefficients case. A rough sketch of the algorithm of the hybrid integral is given below.

Algorithm 2: Hybrid Integral Algorithm

Input: Univariate rational function $f(x) = P(x)/Q(x)$ whose coefficients are known with limited accuracy and represented by floating-point numbers. Here, the approximate GCD of the two input polynomials $P(x)$ and $Q(x)$ is reduced to 1 and $\deg(P) < \deg(Q)$.

Output: Approximate indefinite integral of $f(x)$.

Algorithm:

1. Decompose $f(x)$ into rational and transcendental parts:

$$\int f(x)dx = \frac{s(x)}{t(x)} + \int \frac{p(x)}{q(x)}dx.$$

   In this step, AppGCD is used. If $\mathrm{AppGCD}(Q(x), dQ(x)/dx; \varepsilon) = 1$, then $s(x)/t(x) = 0$.

2. Integrate the transcendental part $p(x)/q(x)$.

   a) Determine all zeros of $q(x)$ by numerical Durand-Kerner method. Since the coefficients of $q(x)$ are real, zeros are limited as real or complex conjugate pairs and written for $m + 2n = \deg(q)$ as real zeros $a_1, a_2, \ldots, a_m$, or complex conjugate zeros $b_1 \pm ic_1, \ldots, b_n \pm ic_n$.

   b) Decompose into a partial fraction

$$\frac{p}{q} = \sum_{k=1}^{m} \frac{e_k}{x - a_k} + \sum_{k=1}^{n} \frac{2(f_k x - b_k f_k - c_k g_k)}{x^2 - 2b_k x + b_k^2 + c_k^2},$$

   where e_k, f_k and g_k are determined by the residue theory as follows:

   - Let $r(x) = p/q'$, where $q' = dq(x)/dx$,
   - for real zeros, $e_k = r(a_k)$;
   - for complex conjugate zeros,

$$f_k = \Re\{r(b_k + ic_k)\} \quad \text{and} \quad g_k = \Im\{r(b_k + ic_k)\},$$

     where $\Re$ and $\Im$ represent real part and imaginary part, respectively.

   c) Substitute two well-known formulas of logarithmic integrals

$$\int \frac{p}{q}dx = \sum_{k=1}^{m} e_k \log \mid x - a_k \mid$$
$$+ \sum_{k=1}^{n} f_k \log \mid x^2 - 2b_k x + b_k^2 + c_k^2 \mid -2g_k \tan^{-1}\left(\frac{x}{c_k} - \frac{b_k}{c_k}\right).$$

Some properties of the algorithm are:

- An *indefinite integral* for a given function with floating-point coefficients is obtained.
- Accurate value of a definite integral is obtained only by substitutions of upper and lower bounds of integral.
- Errors caused by the algorithm are reduced to errors contained in numerical root finding process (Durand-Kerner method) and can be estimated by the Smith theorem.

Results obtained by the hybrid integral algorithm are compared with well-known and widely used numerical integration methods, such as Gaussian quadrature of 32 points (Gauss32), Double-Exponential formula (DE), the Romberg method (Romb) and adaptable Newton-Cotes method (NC). Comparisons are done for three ill-conditioned rational functions:

1. A singularity outside the integral region but close to both ends,
$$I_1 = \int_0^1 \frac{dx}{1000x(x-1) - 0.001}.$$
2. The integrand has a sharp peak in the integral region,
$$I_2 = \int_0^1 \frac{dx}{1000(x-0.5)^2 + 0.001}.$$
3. The integrand has both properties I_1 and I_2,
$$I_3 = \int_0^1 \frac{dx}{x^5 - x^4 - 0.75x^3 + x^2 - 0.25x - 10^{-6}}.$$

Results are shown in Table 1. To obtain accurate value of each integral, I_1, I_2 and I_3, Noda *et al.* [17, 18, 19] used a quadrature by parts. Each integrand which changes rapidly is carefully divided into small but smoothly changing parts and integrated numerically. These values of small integrals are added and accurate result for the integrand is obtained.[1] Results are the same as the results obtained by the hybrid integral algorithm. Thus, it can be said that the hybrid integral algorithm gives better results than well-known numerical methods. Especially it works well for ill-conditioned integrals.

TABLE 1. Comparisons of results of the hybrid integral algorithm and numerical methods

	Hybrid	Numerical Integration			
		G32	DE	Romb	NC
I_1	-0.02763097	-0.01622972	-0.02763097	-0.68482819	5.40966656
I_2	3.13759266	0.19994034	24.6736125	2.98225113	3.13759258
I_3	-5195.24497	-580.39410	-24965.007	-5759.10716	230.75544

The key strategy of the hybrid integral algorithm is to decompose into a partial fraction by computing all zeros of a denominator polynomial of a rational part of the given integrand numerically. For an integral with a parameter or a double integral, the algorithm described above may be easily applied. Here, a denominator of a rational part of an integrand with the bivariate case may be decomposed into a partial fraction by using an approximate factorization algorithm such as [27, 12]. Thus, a kind of *indefinite integral* of the bivariate integral may be obtained.

[1]Current powerful CAS such as Maple 9 or its successor gives the same results and assures the correctness of the hybrid integral algorithm.

4. Hybrid Rational Function Approximation

For rational approximation of a given function or a set of discrete data, the rational interpolation, especially the so-called *naïve rational interpolation*, may be one of the simplest methods. The function is first evaluated at several data points in an interval and changed to the set of discrete data. The set of discrete data is interpolated to a rational function. If the interpolation is done with floating-point computation, pathological features have been observed by Noda *et al.* [18]. To avoid the feature, the AppGCD algorithm is effectively used. Then the rational interpolation approximates a given function or a set of discrete data accurately. We refer to the algorithm as the Hybrid Rational Function Approximation (HRFA). Hereafter, it is simply called HRFA. We introduce HRFA briefly and discuss its theoretical considerations of pathological features.

4.1. Rational Interpolation and Pathological Feature

A rational interpolation is defined as a ratio of numerator and denominator polynomials as

$$r_{m,n}(x) = \frac{p_m(x)}{q_n(x)} = \frac{\sum_{i=0}^{m} a_i x^i}{1 + \sum_{i=1}^{n} b_i x^i} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m}{1 + b_0 x + b_1 x^2 + \cdots + b_n x^n}. \quad (4.1)$$

It interpolates a function $f(x)$ or a set of discrete data on a segment $[x_0, x_{m+n}]$. The rational interpolant (4.1) is called (m, n) *rational interpolant.* Here, the naïve rational interpolation is considered. For $m + n + 1$ $(= N)$ discrete points, $x_0 < x_1 < \cdots < x_{m+n}$, values $f(x_k) = f_k$ $(k = 0, \ldots, m + n)$ are evaluated. Then $m + n + 1$ $(= N)$ simultaneous linear equations

$$\sum_{i=0}^{m} a_i x_k^i - f_k \sum_{i=i}^{n} b_i k_k^i = f_k, \quad k = 0, 1, \ldots, m + n, \quad (4.2)$$

are obtained. The inexact, floating-point coefficients a_i, b_i of the rational interpolant are then determined by solving the system by Gaussian elimination. Noda *et al.* [18] discussed the problem and found that

1. even if a continuous function is interpolated, the denominator of the rational interpolant may have a zero, and the zero causes an *undesired pole*,
2. except for the above zero and pole, the rational interpolant gives accurate approximation of $f(x)$.

Further, the fact that the zero of the numerator polynomial may arise which is very close to the undesired pole has been shown by Kai *et al.* [11] through numerical experiments. The zero and pole, mentioned above, are referred to as *undesired zero and pole.* They also describe the appearance of approximate common factors in the numerator and denominator polynomials. These factors are caused by *undesired zero and pole* and eliminated from the rational interpolant by using HRFA.

The approximately common factor which causes a pathological feature of the rational interpolant is eliminated by using AppGCD. The procedure HRFA ensures high-quality approximation without *undesired zero and pole* for any precision computations. The algorithm of HRFA is shown below. The details of the method and its accuracy are discussed by Noda and Kai [18, 8, 11].

Algorithm 3: HRFA Algorithm

```
    Input: Rational interpolant (naïve rational interpolation)
           $r_{m,n}(x) = \frac{p_m(x)}{q_n(x)}$.
    Output: Reduced rational interpolant without singularities
           $\tilde{r}(x) = \frac{\tilde{p}(x)}{\tilde{q}(x)}$.
  Algorithm:
    1. $\mathrm{AppGCD}(p_m(x), q_n(x)) = g(x)$.
    2. $\tilde{r}(x) = \frac{p_m(x)/g(x)}{q_n(x)/g(x)}$
```

We show the pathological feature of the rational interpolant and how Algorithm 3 works well using an example. Suppose that $r_{4,4}(x)$ is a rational interpolant of the function $f(x) = \log(x+2)$; we obtained the following rational interpolant with a single precision floating-point computation:

$$\begin{aligned} r_{4,4}(x) &\simeq \frac{0.6931 + 214.8599x + 318.6295x^2 + 113.5897x^3 + 9.1269x^4}{1 + 309.2559x + 236.7844x^2 + 48.7819x^3 + 2.1131x^4} \\ &\simeq 4.3195\,\frac{(x+8.77124)(x+2.6710)(x+1)(x+0.0032415)}{(x+16.999)(x+3.8532)(x+2.2286)(x+0.0032416)}\,. \end{aligned}$$

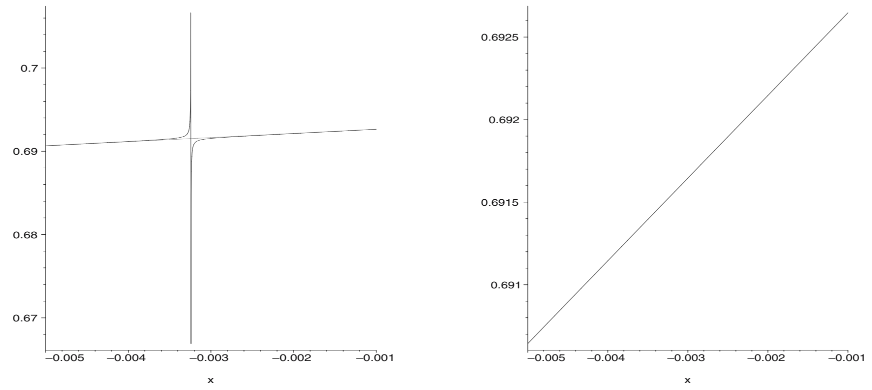
The last terms of the numerator and denominator polynomials, $x + 0.0032415$ and $x + 0.0032416$ respectively, show an approximate common factor. The rational interpolant $r_{4,4}(x)$ causes *undesired zero and pole* by these terms and shown in Fig. 1(a). However, except for a small interval containing *undesired zero and pole*, the rational interpolant constructs an accurate approximation of the function $f(x)$. The HRFA successfully removes the *undesired zero and pole* by taking the AppGCD of $p_m(x)$ and $q_n(x)$. The AppGCD $g(x)$ of the numerator and denominator polynomials of $r_{4,4}(x)$ is computed as

$$g(x) \simeq x + 0.0032423543\,.$$

Then, after dividing $r_{4,4}(x)$ by $g(x)$, a reduced rational function approximation, $R_{\mathrm{HRFA}}(x)$, is obtained as follows:

$$\begin{aligned} R_{\mathrm{HRFA}}(x) &\simeq \frac{9.1269605x^3 + 113.56010x^2 + 318.26127x + 213.82796}{2.1131867x^3 + 48.775051x^2 + 236.62620x + 308.48869} \\ &= 4.3190507\,\frac{(x+8.7712482)(x+2.6710225)(x+0.99999856)}{(x+16.999383)(x+3.8532500)(x+2.2286457)}\,. \end{aligned}$$

The reduced rational function $R_{\mathrm{HRFA}}(x)$ does not have *undesired zero and pole*, and is shown in Fig. 1(b).



(a) $r_{4,4}(x)$ (b) $R_{\mathrm{HRFA}}(x)$

FIGURE 1. Rational function approximation of $\log(x + 2)$ by $r_{4,4}(x)$

4.2. Rational Interpolation and Ill-conditioned Property

Rational functions discussed in the literature [24, 25] are restricted to be irreducible, i.e., the numerator and denominator polynomials have no common factors other than a constant. Litvinov [15] discussed an interpolation of given functions by a rational function and mentioned that the system of linear equations for the rational interpolant turns out to be ill-conditioned in many cases. Although the system itself is ill-conditioned, the fact that the rational interpolant ensures accurate approximation by means of the best approximation is also described. However, Litvinov does not refer to the importance of the appearance of *undesired zero and pole*. On the pathological feature of the rational interpolation, the following facts are shown through numerical experiments by Kai *et al.* [8, 11].

1. An *undesired zero and pole* always appear very close and as a pair.
2. The position of the *undesired zero and pole* changes by degree of the rational function $r_{m,n}(x)$ and by digits of computations.
3. Except for a small interval where the *undesired zero and pole* exist, the rational interpolant gives accurate approximation of a given function or set of data.

Among the above facts, the second refers to the ill-conditioned property of the system of linear equations constructed by the rational interpolation method as mentioned in [15]. The system turns out to be highly ill-conditioned by error propagation during computation and the result in losses in accuracy, i.e., the system (4.2) is sensitive to the precision of computation. Thus, the position of *undesired zero and pole* changes by digits of computation without definite rules. A reason for facts 1 and 3 is discussed below.

From (4.1), we write a system of simultaneous linear equations which determines the coefficients of the rational interpolant, a_i, b_j $(i = 0, \ldots, m; j = 1, \ldots, n)$ as

$$A\mathbf{y} \;=\; B, \quad \mathbf{y} \;=\; (a_0, a_1, \ldots, a_m, b_1, b_2, \ldots, b_n)^T\,, \tag{4.3}$$

where A is a matrix whose size is $N \times N$ for $N = m + n + 1$. A triangular system, $\hat{A}\mathbf{y} \;=\; \hat{B}$, is obtained from (4.3) by Gaussian elimination. Solutions of the system, $\mathbf{y} \;=\; (y_1, y_2, \ldots, y_N)$, are obtained by back substitutions as

$$y_k \;=\; \frac{1}{\hat{A}_{k,k}}\left(\hat{B}_k - \hat{A}_{k,k+1}y_{k+1} - \hat{A}_{k,k+2}y_{k+2}\cdots - \hat{A}_{k,N}y_N\right), \quad k = N-1, \ldots, 1,$$

where $\hat{A}_{i,j}$ and $\hat{B}_j$ denotes the (i, j)-element of $\hat{A}$ and the jth-element of $\hat{B}$, respectively. Since the rational interpolant is ill-conditioned, the resulting $\hat{A}$ may have *bad row*s and become a rank deficient matrix, in many cases. Here, the *bad row* is defined as a row whose elements all take zeros or negligible small values. Thus, in usual computations, it is impossible to obtain $Y_N, Y_{N-1}, \ldots, Y_{N-\gamma+1}$ for $\hat{A}$ whose γ rows are deficient. Then we substitute undetermined symbols $t_1, \ldots, t_\gamma$ as

$$Y_N = t_\gamma, Y_{N-1} = t_{\gamma-1}, \ldots, Y_{N-\gamma+1} = t_1.$$

All elements of $\mathbf{y}$, i.e., the coefficients of the rational interpolant, are represented with undermined symbols. The rational interpolant for the rank deficient case is written as [16]

$$r_{m,n}(x) \;=\; \frac{p_0(x) + t_1p_1(x) + t_2p_2(x) + \cdots + t_\gamma p_\gamma(x)}{q_0(x) + t_1q_1(x) + \cdots + t_\gamma q_\gamma(x)}\,, \tag{4.4}$$

which approximates the given function or a set of discrete data, $f(x)$, accurately.

For the case when system (4.1) is not ill-conditioned, $\hat{A}$ becomes a full rank matrix. In this case, we should take $\gamma = 0$. Then (4.4) is written as

$$r_{m,n}(x) = \frac{p_0(x)}{q_0(x)} \simeq f(x) \text{ with high accuracy.}$$

It is evident that there are no *undesired zero and pole* for the $\gamma = 0$ case. Thus, the appearance of *undesired zero and pole* may be strongly related to $\gamma \geq 1$ cases. That is, the terms depend on the t's in both the numerator and denominator polynomials and may be considered to be a cause of the *undesired zero and pole.* On the other hand, Litvinov discussed a similar problem from a somewhat different viewpoint. In Litvinov's work [15], perturbations of coefficients a_i and b_j are considered. Suppose that a_i and b_j are perturbed and become $a_i + \Delta a_i$ and $b_j + \Delta b_j$. The rational interpolant $r_{m,n}(x) = p_m(x)/q_n(x)$ is perturbed to $\tilde{r}_{m,n}(x) = p_m(x) + \Delta p_m(x)/q_n(x) + \Delta q_n(x)$. Although the rational interpolant has some redundant coefficients, the following relation exists:

$$\frac{p_m(x_i)}{q_n(x_i)} \simeq \frac{\Delta p_m(x_i)}{\Delta q_n(x_i)} \;=\; f(x_i) \tag{4.5}$$

holds for arbitrary i. The function $\Delta p_m(x)/\Delta q_n(x)$ is called *error approximants* [15]. The function, *error approximants*, can also approximate $f(x)$ accurately. The fact suggests that the errors in the numerator and denominator polynomials of the rational interpolant compensate each other. Our $\gamma = 0$ case seems to agree with the $\Delta a_i = \Delta b_j = 0$ case of Litovinov's results.

For understanding the facts more clearly, we show practical examples of a rational approximation of a rational function, i.e., $f(x)$ is taken as a rational function.

4.3. Practical Example for Approximating a Rational Function

Suppose that the rational function $f(x) = 1/(x-3)$ is interpolated by a rational interpolant $r_{m,n}(x)$ in the interval $-1 \le x \le 1$. Here, the results of two types of computations, symbolic computation and numerical computation, are compared.

The case of exact computation

Case $m = 1, n = 1$ Since $r_{1,1} = (a_0 + a_1 x)/(1 + b_1 x)$, the coefficients a_0, a_1 and b_1 should be obtained by solving a system of linear equations (4.2) for a set of discrete data, $x_0 = -1, x_1 = 0, x_2 = 1$. We obtain

$$a_0 = -\frac{1}{3},\ a_1 = 0,\ b_1 = -\frac{1}{3}.$$

Then, $r_{1,1}$ gives an exact interpolation for $f(x)$ as

$$r_{1,1} = \frac{-\frac{1}{3}}{1 - \frac{1}{3}x} = \frac{1}{x-3}.$$

Case $m = 2, n = 2$ A set of discrete data

$$(x_0, x_1, x_2, x_3, x_4) := (-1, -\frac{1}{2}, 0, \frac{1}{2}, 1)$$

is used. After Gaussian elimination, $\hat{A}y = \hat{B}$ shows

$$\begin{bmatrix} 1 & -1 & 1 & -\frac{1}{4} & \frac{1}{4} \\ 0 & 1 & -\frac{2}{3} & \frac{3}{14} & -\frac{5}{14} \\ 0 & 0 & 1 & \frac{1}{14} & \frac{3}{14} \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} \\ -\frac{1}{14} \\ -\frac{1}{42} \\ -\frac{1}{3} \\ 0 \end{bmatrix}.$$

Because the fifth row of $\hat{A}$ and the element $\hat{B}_5$ are zero (*bad row*), we should substitute $b_2 = \hat{B}_5/\hat{A}_{5,5} = 0/0 \Rightarrow t_1$ by an undetermined symbol t_1. The rational interpolant corresponds to the case of $\gamma = 1$. The coefficients of $r_{2,2}$ are then obtained as

$$a_0 = -\frac{1}{3},\ a_1 = t_1,\ a_2 = 0,\ b_1 = -\frac{1}{3} - 3t_1,\ b_2 = t_1.$$

Thus $r_{2,2}$ also interpolates $f(x)$ exactly. A common factor which depends on t_1 which appears in its numerator and denominator. $r_{2,2}$ is written as

$$r_{2,2} = \frac{-\frac{1}{3} + t_1 x}{1 - \frac{1}{3}x - 3t_1 x(1 - \frac{1}{3}x)} = \frac{1 - 3t_1 x}{(x-3)(1-3t_1 x)} = \frac{1}{x-3}.$$

Further, the following facts are shown in $r_{2,2}$:

- terms independent of t_1 give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$.

Case $m = 3, n = 3$ The coefficients of $r_{3,3}$ are obtained from a set of data for $(x_0, x_1, x_2, x_3, x_4, x_5, x_6) := (-1, -\frac{2}{3}, -\frac{1}{3}, 0, \frac{1}{3}, \frac{2}{3}, 1)$. The resulting system $\hat{A}y = \hat{B}$ gives two *bad rows*. Thus the rational interpolant corresponds to the case of $\gamma = 2$ and two undetermined symbols, t_1 and t_2, should be introduced. The coefficients are written as

$$a_0 = -\frac{1}{3}, a_1 = 3t_1 + t_2, a_2 = t_1, a_3 = 0, b_1 = 9t_1 - \frac{1}{3} - 3t_2, b_2 = t_2, b_3 = t_1.$$

After substitutions of these coefficients to $r_{3,3}$, we obtain

$$r_{3,3} = \frac{-\frac{1}{3} + 3t_1 x + t_2 x + t_1 x^2}{(1 - \frac{1}{3}x)(1 - 9t_1 x(1 + \frac{1}{3}) - 3t_2 x)} = \frac{1}{x-3}.$$

The rational interpolant $r_{3,3}$ gives a satisfactory result. A second degree common factor appears in its numerator and denominator. Further, similar to the case of $r_{2,2}$, the following facts are shown:

- terms independent of t_1 and t_2 give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$.

The case of numerical computation

The same computations as in the exact cases are done by floating-point operations. Given a set of discrete data which is the same as above, a function value $f(x_i)$ is evaluated as a floating-point number.

Case $m = 1, n = 1$ Gaussian elimination gives a triangular matrix $\hat{A}$ without *bad row* and we obtain

$$a_0 = -0.33333,\ a_1 = 1.0 \times 10^{-8},\ b_1 = -0.33334$$

as coefficients. Then, the rational interpolant becomes

$$r_{1,1} = \frac{-0.33333}{1 - 0.33334x} \simeq \frac{1}{x-3}\ .$$

The result approximates a given function $f(x)$ accurately. The small coefficient a_1 should be ignored.

Case $m = 2, n = 2$ The results $\hat{A}$ and $\hat{B}$ are obtained after Gaussian elimination as

$$\hat{A} = \begin{bmatrix} 1.0 & -1.0 & 1.0 & -0.25 & 0.25 \\ 0 & 1.0 & -0.66666 & 0.21428 & -0.35714 \\ 0 & 0 & 1.0 & 0.071428 & 0.21428 \\ 0 & 0 & 0 & 1.0 & 3.0 \\ 0 & 0 & 0 & 0 & -0.7 \times 10^{-9} \end{bmatrix}, \ \hat{B} = \begin{bmatrix} -0.25 \\ -0.071428 \\ -0.023809 \\ -0.33333 \\ -0.3 \times 10^{-9} \end{bmatrix}.$$

Different from the exact computation case, the last row of $\hat{A}$ and $\hat{B}$ contain very small values. They may be caused by errors of floating-point computations. Straightforward numerical computations give the coefficients of $r_{2,2}$ as

$$a_0 = -0.33333, a_1 = 0.43611, a_2 = -2.0 \times 10^{-9}, b_1 = -0.975, b_2 = -0.43611.$$

Although all coefficients are not reliable, the above coefficients give the rational interpolant

$$r_{2,2} = \frac{-0.33333 + 0.43611x}{1 - 0.975x - 0.43611x^2} \simeq \frac{-0.43611(x + 0.76433)}{0.43611(x-3)(x+0.76433)} \simeq \frac{1}{x-3}.$$

The result is satisfactory. Approximate common factors appear in the numerator and denominator of $r_{2,2}$. If we apply HRFA to the above $r_{2,2}$, then approximate common factors are reduced.

Case $m = 3, n = 3$ Similar computations with the above elements in two rows of $\hat{A}$ and $\hat{B}$ become very small and depend on the environments of computers. However, the results are always meaningful and similar to the case of $m = n = 2$. A set of coefficients obtained with one computation is

$$\begin{aligned} a_0 &= -0.33333,\ a_1 = -0.72881,\ a_2 = 0.17352,\ a_3 = 0.29 \times 10^{-7}, \\ b_1 &= 1.8530,\ b_2 = -1.2493,\ b_3 = 0.17352. \end{aligned}$$

The resulting rational interpolant is obtained as

$$r_{3,3} \simeq \frac{0.17352(x^2 - 4.2x - 1.9207)}{0.17352(x-3)(x^2 - 4.2x - 1.9209)} \simeq \frac{1}{x-3}.$$

Approximate common factors which are the second order polynomial appear in the numerator and denominator of $r_{3,3}$. They may cause an *undesired zero and pole* of the rational interpolant but this is reduced by HRFA.

4.4. Facts Obtained from Practical Examples

If we wish to approximate a given function or a set of data accurately, we should increase the number of evaluation points in an interval. If the function can be written as a lower degree rational function, then there are many redundant points which could occur. They may cause *bad rows* in the triangular matrix. If undetermined symbols are substituted, then we can obtain all coefficients of the rational interpolant as shown in (4.4). However, in the approximate floating-point computation, the above computations for $f(x) = 1/(x-3)$ show some interesting results in terms of the appearance of pathological features of the rational interpolant. They are summarized as follows.

Exact computation

- In the case $\gamma = 0$, p_0/q_0 represents $f(x)$ exactly.
- In the case $\gamma = 1$, there is a bad row in the triangular system. An undetermined symbol t_1 should be introduced. The rational interpolant shows
$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = f(x). \tag{4.6}$$
- In the case $\gamma = 2$, two undetermined symbols t_1 and t_2 should be introduced. The rational interpolant shows
$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = \frac{p_2}{q_2} = f(x). \tag{4.7}$$

Approximate floating-point computation

- Instead of the introduction of undetermined symbols, negligible small values appear in the triangular system. Coefficients of the rational interpolant are obtained by straightforward numerical computations. Results of the interpolant contain approximate common factors in its numerator and denominator.
- The *undesired zero and pole* are caused by the above approximate common factors and are removed by the HRFA algorithm.

The above facts, especially (4.6) and (4.7), suggest an interesting relation
$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = \frac{p_2}{q_2} = \cdots = \frac{p_\gamma}{q_\gamma} = f(x)$$
for the case of exact computation. It also agrees with Litvinov's result (4.5) for floating-point computation.

Next we consider the case of applications of HRFA to functions or a set of data which is not a rational function but a general continuous function such as a logarithmic function. As already discussed in this section, when we approximate $f(x) = \log(x + 2)$ by a rational function, we know the appearance of an approximate common factor in the numerator and denominator polynomials in the rational interpolant. By the approximate common factor, the pathological feature, *undesired zero and pole*, of the rational function appears. As future theoretical work, it is important to discuss much more the *undesired zero and pole* of the HRFA algorithm. In any event, in almost all cases, HRFA gives accurate rational approximation for a given function or a set of data.

5. Practical Applications of Hybrid Algorithms

Practical applications and developments of the HRFA and AppGCD hybrid algorithms are considered here. They are (1) smoothing data containing small and/or large errors and (2) obtaining the Cauchy principal value integral. In the former, we can show the *robustness* of our method by comparisons with traditional methods such as the least square method. In the latter, we use the hybrid integral algorithm in which the AppGCD is effectively used [18].

5.1. Data Smoothing by HRFA

We show how HRFA works well for data including small errors and also large errors such as input errors. A set of data which includes some types of error is approximated by a naïve rational interpolation. Here, a set of data with large errors is considered as an *unattainable* point and causes an approximate GCD of the numerator and denominator polynomials of the rational function. Thus the error is removed from the reduced rational function. An early work of our method is given in [8].

5.1.1. Method for Data Smoothing. Two cases, (1) small error (noise) and (2) large error (experimental error), are considered. Let the input be a set of discrete data

$$D = \{(x_i, f_i) \mid i = 1, \dots, m+n+1\}.$$

Small error case: Almost all the (x_i, f_i)'s are approximated by a rational function but with a small error. If we compute the approximate GCD of numerator and denominator polynomials with relatively large ε, we obtain a lower degree rational function by the approximate GCD. The resulting rational function gives a smooth function which approximates the given data D.

Large error case: We show how HRFA is effective for smoothing a set of data including a large error. The rational interpolation is shown as $r_{m,n}(x) = p_m(x)/q_n(x)$ for D. The rational function is written as

$$q_n(x_i) f_i = p_m(x_i), \quad i = 1, \dots, m+n+1.$$

Thus, if $q_n(x_i) \neq 0$ for $i = 1, \dots, m+n+1$, then $p_m(x)$ is uniquely determined. On the other hand, if $q_n(x_k) = 0$ is satisfied for a $k \in [1, m+n+1]$, then $p_m(x_k)$ vanishes also. Further, in this case, the following expression may be satisfied:

$$\lim_{x \to x_k} \frac{p_m(x)}{q_n(x)} \neq f_k.$$

The rational function $r_{m,n}$ is not through a point (x_k, f_k) in D. We call this point an *unattainable* point. Further, it is easy to say that if (x_k, f_k) is the *unattainable* point, then $p_m(x)$ and $q_n(x)$ have a common factor at $x = x_k$. With detailed considerations, we write the above facts as the following theorem.

Theorem. *Let s and t be positive integers satisfying $s+t<m+n$. Further let $r_{s,t}=p_s(x)/q_t(x)$ where p_s and q_t are polynomials with degrees s and t, respectively. A data set D is divided into two groups*

$$\begin{aligned} D_1: &\quad \{(x_{j_i}, f_{j_i}) \mid i = 1, \dots, max(m+t, n+s)+1\},\\ D_2: &\quad \{(x_{k_i}, f_{k_i}) \mid i = 1, \dots, \min(m-s, n-t)\}, \end{aligned}$$

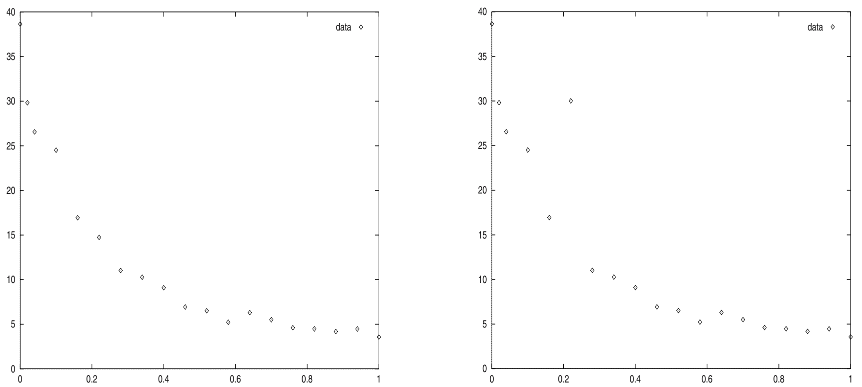
where $\{x_1, \dots, x_{m+n+1}\} = \{x_{j_1}, x_{j_2}, \dots, x_{k_1}, x_{k_2}, \dots\}$. If the relations

$$\begin{aligned} f_{j_i} &= r_{s,t}(x_{j_i}), \quad i = 1, \dots, max(m+t, n+s)+1,\\ f_{k_i} &\neq r_{s,t}(x_{k_i}), \quad i = 1, \dots, \min(m-s, n-t), \end{aligned}$$

hold, then the points in D_2 are unattainable points.

The theorem may be easily proved by *contradiction*. Suppose that $r_{m,n}(x) = p_m(x)/q_n(x)$ is a rational function that interpolates the given set of discrete data and is not the same as the rational function $r_{s,t} = p_s(x)/q_t(x), s < m, t < n$ described in the theorem above. If a relation $r_{m,n}(x) = r_{s,t}(x)$ holds, then the numbers of solutions of the relation are at most $\max(m+n, n+s)$. However, there are $\max(m+t, n+s)+1$ points which should be on $r_{s,t}$. Thus, the assumption that $r_{m,n}(x) \neq r_{s,t}(x)$ leads to a contradiction. This shows that $r_{m,n}(x) = r_{s,t}(x)$.

5.1.2. Examples of Data Smoothing. Next, we show with some examples how our method works well for a given set of data with small and large errors. We consider here two given types of input experimental data with small error as shown in Fig. 2(a) and large error as shown in Fig. 2(b).



(a) Including small error (b) Including large error

FIGURE 2. Given an experimental set of 19 discrete data

Small error case: Since the given set of discrete data consists of 19 points ($m+n+1 = 1$), it is interpolated by a rational function $r_{9,9}(x) = p_9(x)/q_9(x)$.

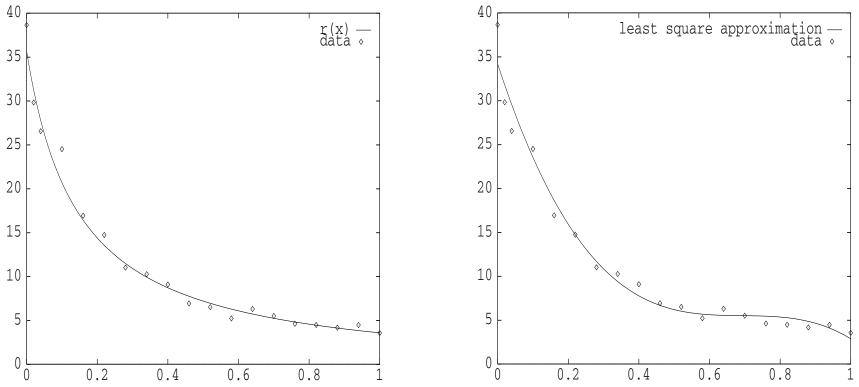
Following the above method, the AppGCD of both polynomials is computed as

$$\begin{aligned} g_8(x) &= \mathrm{AppGCD}(p_9(x), q_9(x), 0.1) \\ &= -1.991\times10^{-5} + 6.532\times10^{-3}x - 0.1549x^2 + 1.400x^3 - 6.367x^4 \\ &\quad +15.99x^5 - 22.43x^6 + 16.41x^7 - 4.862x^8. \end{aligned}$$

The result of our method is obtained in dividing $r_{9,9}(x)$ by the AppGCD as

$$\tilde{r}_{1,1}(x) = \frac{35.66 - 7.560x}{1.000 + 6.871x}.$$

The result $\tilde{r}_{1,1}(x)$ is shown in Fig. 3(a) and compared with the result of the least square method for the same set of discrete data shown in Fig. 3(b).



(a) $\tilde{r}_{1,1}(x)$ (b) Least square approximation

FIGURE 3. Results of our method and the least square method

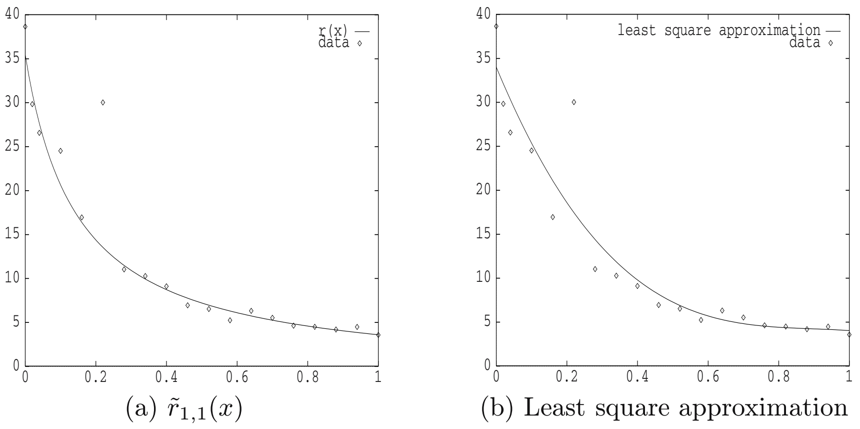
Large error case: The main parts of the given set of discrete data are the same as in the small error case. The data smoothing method is the same as above as well. We also obtain the AppGCD whose degree is 8 from a rational function $r_{9,9} = p_9(x)/q_9(x)$ as

$$\begin{aligned} g_8(x) &= \mathrm{AppGCD}(p_9(x), q_9(x), 0.1) \\ &= -1.558\times10^{-5} + 6.067\times10^{-3}x - 0.1460x^2 + 1.339x^3 - 6.167x^4 \\ &\quad +15.65x^5 - 22.13x^6 + 16.29x^7 - 4.850x^8. \end{aligned}$$

The result of our method is obtained in dividing $r_{9,9}(x)$ by the AppGCD as

$$\tilde{r}_{1,1}(x) = \frac{35.43 - 7.532x}{1.000 + 6.815x}.$$

The result $\tilde{r}_{1,1}(x)$ is shown in Fig. 4(a) and compared with the result of the least square method for the same set of discrete data shown in Fig. 4(b).



(a) $\tilde{r}_{1,1}(x)$ (b) Least square approximation

FIGURE 4. The results of our method and the least square method

The above results of data smoothing for experimental data with small/large error show the possibility of our method based on HRFA as one of the powerful methods for data smoothing.

5.2. Cauchy Principal Value Integral by Hybrid Methods

We apply HRFA to the Cauchy Principal Value integral (CPV) and a kind of the Cauchy-type singular integral equation. Through examples, we show how computations by HRFA give more accurate and stable results than usual numerical computation. Detailed discussions are in [9, 10].

The CPV is defined as

$$\wp\int_a^b \frac{f(x)}{x-\lambda}dx = \lim_{\varepsilon\to 0+}\left(\int_a^{\lambda-\varepsilon}\frac{f(x)}{x-\lambda}dx + \int_{\lambda+\varepsilon}^b \frac{f(x)}{x-\lambda}dx\right).$$

The difficulty on the numerical evaluation of such integral is that the integrand $\frac{f(x)}{x-\lambda}$ is unbounded in the neighborhood of $x=\lambda$. The Hilbert transform of $f(x)$ is a well-known method of subtracting the singularity as

$$\wp\int_a^b \frac{f(x)}{x-\lambda}dx = \int_a^b \frac{f(x)-f(\lambda)}{x-\lambda}dx + \wp\int_a^b \frac{f(\lambda)}{x-\lambda}dx. \tag{5.1}$$

A lot of methods have been proposed for numerical evaluation of the CPV, when the singular point λ is known before the evaluation. Most of the methods are based on function approximations such as polynomial approximation and spline approximation. However, it becomes difficult to evaluate the CPV by traditional numerical methods for the case when the integrand is complex or given by experimental data. We propose an efficient approach to evaluate the CPV by a straightforward application of HRFA as follows:

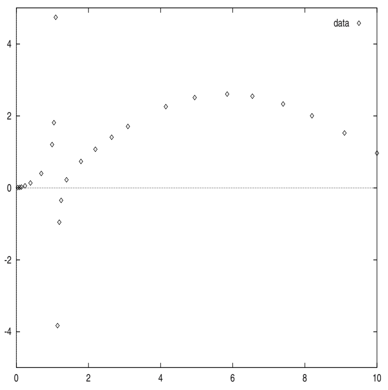


FIGURE 5. Given data from the analysis of optical waveguides

1. Obtain a set of data $D = (x_i, f(x_i)/(x_i - \lambda))$, $i = 0, \ldots, m+n$, where x_i and $f(x_i)/(x_i - \lambda)$ show the control variable and the associated value of the integrand, respectively.
2. Approximate given points and values to a rational function $r(x)$ with HRFA.
3. Obtain approximate indefinite integral of $r(x)$ by the hybrid integral algorithm.

We consider an example of using this approach. The experimental data from a problem on the analysis of optical wavewguides are shown in Fig. 5. In this case, the location of the pole is not known a priori. Thus numerical methods are difficult to apply. Naïve rational interpolation $r_{12,12}(x)$ which has *undesired zero and pole* in the interval $[0, 10]$ is first obtained. Then the HRFA algorithm reduces it to $\hat{r}_{9,9}(x)$ whithout these *zero and pole* for the accuracy $\varepsilon = 10^{-4}$ [8]. We can obtain the approximate indefinite integral, I, of $\hat{r}_{9,9}(x)$ in $[0, 10]$ by the hybrid integral algorithm. The value of the CPV is obtained by estimations of I at $x = 10$ and $x = 0$ as

$$I(x = 10) - I(x = 0) \simeq 16.8932.$$

The result is consistent with the experimental results concerning optical waveguides. Many physical/engineering problems are described by equations involving the CPV and given by discrete experimental data. Traditional numerical methods and methods using Hilbert transform cannot be applied to thse problems. Our method, the hybrid integral algorithm using HRFA, can give approximate but stable results for such problems.

The method is naturally extended to solve a Cauchy-type singular integral equation. The equation is written as

$$ag(x) + \frac{b}{\pi}\int_{-1}^{1}\frac{g(t)dt}{t-x} + \lambda\int_{-1}^{1}k(x,t)g(t)dt = f(x), \quad -1 < x < 1. \tag{5.2}$$

The so-called dominant equation, a special case of (5.2),

$$ag(x) + \frac{b}{\pi}\int_{-1}^{1}\frac{g(t)dt}{t-x} = f(x), \quad -1 < x < 1,$$

is considered in [10]. The result is compared with that obtained by other numerical methods, such as the Lobatto-Chebyshev scheme. The resulting value of our method agrees but requires smaller interpolation points than other methods.

6. Conclusion and Future Work

In this paper, hybrid symbolic-numeric computation and its applications are described. We show how hybrid computation which first began with the introduction of the notion of approximate GCD (AppGCD) works well for obtaining accurate results for many kinds of ill-conditioned problems. It can be applied to most problems which appear in scientific/engineering computations. The variables treated in them are inexact and are represented by floating-point numbers. The hybrid rational function approximation (HRFA) algorithm in which AppGCD is effectively used has theoretical implications and many applications. Regarding its theoretical implications, a pathological feature, the appearance of *undesired zero and pola*, is considered through practical examples. The fact that this may be caused by the ill-conditioned property of a system of simultaneous linear equations which determines all coefficients of the rational interpolant is mentioned. On the other hand, as applications of HRFA, several problems such as smoothing a set of data which contains small and/or large errors, integrating ill-conditioned functions or a set of data (hybrid integral) and also applying the Cauchy principal value integral are discussed.

The discussions herein are summarized as follows.

- AppGCD forms the basis of today's developments of hybrid symbolic-numeric computations.
- A hybrid integral algorithm, which is an extension of Horowitz' algorithm to an integrand with limited accuracy or perturbed within small tolerance, i.e., in the inexact case, works well for obtaining definite/indefinite integral of ill-conditioned rational functions.
- HRFA gives accurate rational approximations for a given function or a set of discrete data. An ill-conditioned property of the system of simultaneous linear equations which determines the coefficients of the rational interpolant causes the pathological feature, *undesired zero and pole*. However, the feature is eliminated by the AppGCD of numerator and denominator polynomials.
- A set of discrete data including small and/or large error is well smoothed by HRFA. Here, errors containing input data correspond to *unattainable* points and are removed by HRFA. The results of data smoothing show the *robustness* of our method.

- HRFA is used for computing the Cauchy principal value integral (CPV). Different from many proposed numeric methods for CPV, it is not necessary to have a priori knowledge of the position of singularity.

The following problems are important for further developing research related to hybrid computations.

1. Apply hybrid computation to practical engineering problems in a wider area, such as control theory.
2. Make clear the relation between hybrid computation and ill-conditioned problems mathematically such as the appearance of the pathological feature of the HRFA algorithm.
3. Apply the hybrid integral algorithm to obtain a kind of "indefinite" integral of a rational function with a parameter or a double integral.

Finally, the author wishes to acknowledge his collaborator Prof. Tateaki Sasaki of the University of Tsukuba and many graduate students who worked hard to establish the basis of hybrid computation during their studies at Ehime University. Especially, among them, Mr. Masaaki Ochi, Mr. Ei-ichi Miyahiro and Ms. Yumi Takashima, formerly Murakami, developed studies and obtained important results of Sects. 2, 3 and 4 of this paper, respectively. Further, Dr. Hiroshi Kai who is also one of them has been involved with almost all parts of the research.

References

[1] W. Auzinger and H. J. Stetter: An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations, *Numerical Mathematics*, **86**, ISNM, edited by R. P. Agarwal *et al.*, Birkhäuser, pp. 11–30, 1988.

[2] E. H. Bareiss: The numerical solution of polynomial equations and the resultant procedure, *Mathematical Method for Digital Computers*, vol. 2, John Wiley, pp. 185–214, 1967.

[3] R. M. Corless, P. M. Gianni, B. M. Trager and S. M. Watt: The singular value decomposition for polynomial systems, *Proc. ISSAC '95*, ACM Press, pp. 195–207, 1995.

[4] J. Grabmeier, E. Kaltofen and V. Weispfenning: *Computer Algebra Handbook*, Springer, pp. 112–125, 2003.

[5] I. Z. Emiris, A. Galligo and H. Lombardi: Certified approximate univariate GCDs, *J. Pure Appl. Algebra*, **117**, pp. 229–251, 1997.

[6] E. Horowitz: Algorithms for partial fraction decomposition and rational function integration, *Proc. 2nd ACM Symp. Symbolic and Algebraic Manipulation*, pp. 441–457, 1971.

[7] V. Hribernig and H. J. Stetter: Detection and validation of clusters of polynomial zeros, *J. Symb. Comp.*, **24**, pp. 667–681, 1997.

[8] H. Kai and M.-T. Noda: Hybrid rational function approximation and data smmoothing, *J. Jap. Appl. Math.*, **3**, pp. 323–336, 1993 (in Japanese).

[9] H. Kai and M.-T. Noda: Cauchy principal value integral using hybrid integral, *SIGSAM BULLETIN*, **31**, pp. 37–38, 1997.

[10] H. Kai and M.-T. Noda: Hybrid computation of Cauchy-type singular integral, *SIGSAM BULLETIN*, **32**, pp. 59–60, 1998.

[11] H. Kai and M.-T. Noda: Hybrid rational function approximation and its accuracy analysis, *Reliable Computing*, **6**, pp. 429–438, 2000.

[12] E. Kaltofen: Fast parallel absolute irreducibility testing, *J. Symb. Comp.*, **1**, pp. 57–67, 1985.

[13] E. Kaltofen: http://www.math.ncsu.edu/˜kaltofen/bibliography/kaltofen.html#GKW02, pp. 112–129, 2005.

[14] N. Karmarkar and Y. N. Lakshman: Approximate polynomial greatest common divisors and nearest singular polynomials, *Proc. ISSAC '96*, ACM Press, pp. 35–39, 1996.

[15] G. Litvinov: Approximate construction of rational approximations and the effect of error autocorrection: Applications, *Russian Journal of Mathematical Physics*, vol. 1, no. 3, pp. 1–45, 1994.

[16] Y. Nakajima, H. Kai and M.-T. Noda: Hybrid rational function approximation and ill-conditioned problem, *J. JSSAC*, **11**, pp. 141–152, 2005.

[17] M.-T. Noda and E. Miyahiro: On the symbolic/numeric hybrid integration, *Proc. ISSAC '90*, ACM Press, p. 304, 1990.

[18] M.-T. Noda, E. Miyahiro and H. Kai: Hybrid rational function approximation and its use in the hybrid integration, *Advances in Computer Methods for Partial Differential Equations VII*, edited by R. Vichnevetsky, D. Knight and G. Richter, IMACS, pp. 565–571, 1992.

[19] M.-T. Noda and E. Miyahiro: Extension of the hybrid integral by using the hybrid rational function approximation, *J. Jap. Appl. Math.*, **2**, pp. 193–206, 1992 (in Japanese).

[20] M.-T. Noda and T. Sasaki: The interval arithmetic for the ill-conditioned polynomial equation, *RIMS* (Research Institute of Mathematical Sciences, Kyoto University) *Lecture Note*, **673**, pp. 47–61, 1988.

[21] M.-T. Noda and T. Sasaki: Approximate GCD and its application to ill-conditioned algebraic equations, JCAM, **38**, pp. 335–351, 1991.

[22] M. Ochi, M.-T. Noda and T. Sasaki: Approximate GCD of multivariate polynomials and application to ill-conditioned system of algebraic equations, *J. Inf. Proces.*, **14**, pp. 292–300, 1991.

[23] V. Y. Pan: Computation of approximate polynomial GCDs and an extension, *Proc. 9th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 68–77, 1998.

[24] J. R. Rice: *The Approximation of Functions* II, Addison-Wesley, pp. 76–122, 1969.

[25] T. J. Rivlin: *An Introduction to Approximation of Functions*, Blaisdell, pp. 120–141, 1969.

[26] T. Sasaki and M.-T. Noda: Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations, *J. Inf. Proces.*, **12**, pp. 159–168, 1989.

[27] T. Sasaki and M. Sasaki: A unified method for multivariate polynomial factorization, *J. Inf. Proces.*, **10**, pp. 21–39, 1993.

[28] A. Schönhage: Quasi-GCD computations, *J. Complexity*, **1**, pp. 118–137, 1985.

[29] T. W. Sederberg and G. Z. Chang: Best linear common divisors for approximate degree reduction, *Computer-Aided Design*, **25**, pp. 163–168, 1993.

[30] K. Shirayanagi and M. Sweedler: A theory of stabilizing algebraic algorithms, Tech. Rep. 95-28, Cornell Univ., pp. 1–92, 1995.

Matu-Tarow Noda
Center for Information Technology
Ehime University
Ehime Campus Information Service
Co. Ltd., Bunkyo-cho 3
Matsuyama, 790-8577, Japan
e-mail: `noda@ecis.co.jp`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 47–53
© 2007 Birkhäuser Verlag Basel/Switzerland

Rational Interpolation and Its Ill-conditioned Property

Hiroshi Kai

Abstract. A rational interpolation is obtained by solving a system of linear equations. However, when the system is solved by floating point arithmetic, there appears a pathological feature such as undesired zeros and poles. In this paper, a method is described with the help from computer assisted proof to eliminate the feature.

Keywords. Approximate GCD, rational interpolation, ill conditioned property.

1. Introduction

A rational interpolation approximates a given function to a rational function, which is defined as a ratio of numerator and denominator polynomials as

$$r_{m,n}(x) = \frac{p_m(x)}{q_n(x)} = \frac{a_0 + a_1 x + \cdots + a_m x^m}{1 + b_1 x + \cdots + b_n x^n}. \tag{1.1}$$

This interpolates a function $f(x)$ or a set of discrete data on a range $[\alpha, \beta]$. For any given points $\alpha < x_0 < x_1 < \cdots < x_{m+n} < \beta$ we have

$$r_{m,n}(x_k) = f(x_k) := f_k, \quad \text{for} \quad k = 0, \ldots, m+n.$$

Then we obtain the linearized equations

$$\sum_{i=0}^{m} a_i x_k^i - f_k \sum_{j=1}^{n} b_j x_k^j = f_k. \tag{1.2}$$

The unknown coefficients a_i, b_j can be determined by solving the Equ. (1.2) by Gaussian elimination.

However, when the linearized equations are solved by floating point arithmetic, there appears a pathological feature such as undesired zeros and poles [6]. A reason of the appearance of undesired zeros and poles is highly ill-conditioned property of the Equ. (1.2) [4, 5].

For the well-conditioned case, there are modern fast algorithms for rational interpolation (see, e.g., [8]). When we consider only the ill-conditioned case, the naïve method such as Gaussian elimination may be investigated.

In this paper, the reason of the appearance of undesired zeros and poles is stated more precisely. A method to eliminate the pathological feature is presented using computer assisted proof.

2. Undesired Zeros and Poles

Suppose that the system Equ. (1.2) is $Ay = B$. Here, the coefficients a_i, b_j are represented by a vector $y \in R^{m+n+1}$ as $y = (a_0, \ldots, a_m, b_1, \ldots, b_n)^T$. Then the matrix $A \in R^{(m+n+1)\times(m+n+1)}$ and the vector $B \in R^{m+n+1}$ are represented as follows:

$$A = \begin{pmatrix} 1 & x_0 & \cdots & x_0^m & -f_0 x_0 & \cdots & -f_0 x_0^n \\ 1 & x_1 & \cdots & x_1^m & -f_1 x_1 & \cdots & -f_1 x_1^n \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m+n} & \cdots & x_{m+n}^m & -f_{m+n} x_{m+n} & \cdots & -f_{m+n} x_{m+n}^n \end{pmatrix}, \tag{2.1}$$

$$B = (f_0, f_1, \ldots, f_{m+n})^T . \tag{2.2}$$

The matrix A containing two column blocks of Krylov matrix is known as ill-conditioned matrix [1]. The ill-conditioned property of the system gives the pathological feature such as the appearance of approximate common factors. The approximate common factors cause undesired zeros and poles in the rational interpolants. We analyze the system in the following two cases:

1. the matrix A is a singular matrix,
2. the matrix A is an ill-conditioned matrix.

2.1. Case 1: A Is a Singular Matrix

Suppose that the given function $f(x)$ is a rational function $r_{M,N}(x) = p_M(x)/q_N(x)$, where $M < m$ and $N < n$; then the matrix A will be singular. In order to show this, we need the following property of the rational interpolation.

Lemma 2.1. *Suppose that $r_{m,n}(x)$ is a rational interpolant of the rational function $r_{M,N}(x)$. Then*

$$r_{m,n}(x) \sim r_{M,N}(x). \tag{2.3}$$

Proof. If we assume $r_{m,n}(x) \neq r_{M,N}(x)$, then the number of intersection of these functions is up to $\max\{M + n, N + m\}$. However we have $m + n + 1$ points to interpolate $r_{M,N}(x)$. This leads to contradiction of the assumption $r_{m,n}(x) \neq r_{M,N}(x)$. □

Lemma 2.1 implies that $r_{m,n}(x)$ is represented by the following equation:

$$r_{m,n}(x) = \frac{g(x)p_M(x)}{g(x)q_N(x)}, \tag{2.4}$$

where $g(x)$ is a polynomial and the degree is $\gamma := \min\{m - M, n - N\}$. From this observation, the number of the unknown coefficients for this problem is reduced to $m + n + 1 - \gamma$. Thus, the following property holds immediately.

Corollary 2.2. *The matrix A is singular and its rank is $m + n + 1 - \gamma$.*

Suppose that $\hat{A}y = \hat{B}$ is a triangularized system by Gaussian elimination of $Ay = B$. The last γ rows of $\hat{A}$ and $\hat{B}$ will be all zero. We may substitute symbols $t_1, \ldots, t_\gamma$ in the solution to have a unique solution, such as $b_n = t_\gamma, \ldots, b_{n-\gamma-1} = t_1$. Then, we have the following form:

$$r_{m,n}(x) = \frac{p_0(x) + t_1p_1(x) + \cdots + t_\gamma p_\gamma(x)}{q_0(x) + t_1q_1(x) + \cdots + t_\gamma q_\gamma(x)}. \tag{2.5}$$

Since $r_{M,N}(x)$ is free from the symbols $t_1, \ldots, t_\gamma$, $p_M(x) = p_0(x)$ and $q_N(x) = q_0(x)$. From Lemma 2.1, $p_i(x) = v_i(x)p_0(x)$ and $q_i(x) = v_i(x)q_0(x)$ must be allowed as well, where $v_i(x)$ are polynomials. Hence, the following property holds.

Theorem 2.3 (Murakami et al. [5]). *If A is singular (has the rank $m+n+1-\gamma$), then $r_{m,n}(x)$ may be represented as follows:*

$$r_{m,n}(x) = \frac{g(x)p_M(x)}{g(x)q_N(x)}, \tag{2.6}$$

$$g(x) = \begin{cases} 1 + \sum_{i=1}^{\gamma} t_iv_i(x) & \text{for } \gamma > 0, \\ 1 & \text{for } \gamma = 0. \end{cases} \tag{2.7}$$

The converse may also hold.

Example. For $\gamma = 1$, if $r_{M,N}(x)$ is represented as

$$r_{M,N}(x) = \frac{a_0 + \cdots + a_Mx^M}{1 + b_1x + \cdots + b_Nx^N}, \tag{2.8}$$

then Theorem 2.3 shows that $g(x) = 1 + t_1v_1(x)$, where $v_1(x) = x/b_M$ by easy derivation.

The rational interpolant $r_{m,n}(x)$ may be computed in floating point arithmetic, but the symbols t_i will be substituted by rounding errors. This computation will raise an approximate GCD in $r_{m,n}(x)$.

2.2. Case 2: A Is an Ill-conditioned Matrix

If the given function $f(x)$ is not a rational function, the matrix A is not singular in general. But, there are undesired zeros and poles in the rational interpolant, as we have observed experimentally in [3] and [6]. As is well-known in numerical computation, if the condition number of A is large, then the matrix A is numerically singular.

Example. We compute a rational interpolant $r_{4,4}(x)$ for the function $f(x) = \log(x+2)$ in $x \in [0,1]$. By using the similar procedures discussed in the previous section, $\hat{A}$ and $\hat{B}$ are obtained after Gaussian elimination as

$$\begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & & 1 & -1.1 & -1.1 & -1.1 & -1.1 \\ 0 & & -0.4 & 0.09 & 0.3 & 0.4 & 0.5 \\ 0 & & 0.08 & 0.004 & -0.02 & -0.06 & -0.1 \\ 0 & \vdots & -0.03 & 0.0005 & -0.002 & 0.008 & 0.04 \\ 0 & & 0 & 0.0002 & -0.00007 & 0.0002 & -0.001 \\ 0 & & 0 & 0 & -0.0000007 & 0.000006 & -0.00007 \\ 0 & & 0 & 0 & 0 & 0.2\cdot 10^{-7} & -0.7\cdot 10^{-6} \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0.5\cdot 10^{-7} \end{bmatrix}$$

and

$$\left(0.7,\ 0.4,\ 0.02,\ 0.001,\ 0.0002,\ 0.000009,\ -0.00000006,\ 0.5\cdot 10^{-9},\ -0.3\cdot 10^{-9}\right)^T$$

by 9 digits floating point arithmetic. More precisely, the elements $\hat{A}_{9,9} \simeq 0.52292 \cdot 10^{-7}$ and $\hat{B}_9 \simeq -0.27990 \cdot 10^{-9}$ in the last row are very small. By a straightforward computation, we can obtain the following rational interpolant:

$$r_{4,4}(x) \ \simeq\ 4.5843 \frac{(x+11.392)(x+2.9853)(x+1.0001)(x-0.8305033948)}{(x+21.676)(x+4.4851)(x+2.3139)(x-0.8305033945)}\ .$$

We can observe an undesired zero and pole appearing around 0.83053 in $r_{4,4}(x)$.

Here, we eliminate the undesired zero and pole from $r_{4,4}(x)$. Then the remaining part is as follows:

$$\begin{aligned} r_{3,3}(x) &= 4.5843 \frac{(x+11.392)(x+2.9853)(x+1.0001)}{(x+21.676)(x+4.4851)(x+2.3139)} \\ &= \frac{0.69315 + 0.98608x + 0.31337x^2 + 0.020379x^3}{1 + 0.70126x + 0.12658x^2 + 0.00444539x^3}. \end{aligned}$$

In order to show that A is numerically singular, let us assume that the given function $f(x)$ is this rational function $r_{3,3}(x)$. From Theorem 2.3, since $t_1 = \hat{B}_9/\hat{A}_{9,9} = -0.27990 \cdot 10^{-9}/0.52292 \cdot 10^{-7} = -0.00535265$ and $b_3 = 0.00444539$, the GCD is constructed as follows:

$$g(x) = 1 + \frac{t_1}{b_3}x \simeq 1 + \frac{-0.00535265}{0.00444539}x = -1.20409(x - 0.830503).$$

This result agrees with the position of undesired zero and pole. This shows that the matrix A is numerically singular.

3. To Eliminate the Feature

There exists a well-known property to state how much a regular matrix is close to a singular matrix. Let P be the set of singular matrices. Let $\mathrm{dist}(A, P)$ denote the minimum distance from the matrix A to the set P. Then

$$\mathrm{dist}(A, P) = \| A^{-1} \|^{-1} = \frac{\| A \|}{\mathrm{cond}(A)},$$

where $\| \cdot \|$ denotes an arbitrary norm and $\mathrm{cond}(A)$ is the condition number of the matrix A (see, e.g., [2]).

This property indicates that we need higher digits of floating point arithmetic than the following positive integer d to eliminate the feature:

$$d = \left\lceil -\log_{10} \frac{\| A \|}{\mathrm{cond}(A)} \right\rceil.$$

In the example for $\log(x + 2)$, the condition number of A is estimated as

$$\frac{||A||}{\mathrm{cond}A} = 6.974 \cdot 10^{-12}.$$

This result shows that we need $d = 12$ digits at least to have an accurate result for this problem. Thus, the matrix A is numerically singular in 9 digits floating point arithmetic and, in fact, we have an approximate GCD $g(x) = x - 0.830503$.

We can consider two methods to eliminate the pathological feature:

1. use approximate GCD and eliminate the undesired zeros and poles, or
2. compute in higher precision.

The first method is called Hybrid Rational Function Approximation (HRFA) and works well for computing a rational approximation in limited precision [3, 6]. If higher precision is available, the second method may give us an accurate rational interpolant for a function.

To verify the solutions of the linear equations $Ay = B$, we refer to Algorithm 3.1 in [7]. It can verify nonsingularity of A, after an approximate inverse matrix of A is computed. If the verification is successful, then the result is guaranteed to be accurate.

However, if the condition number of A is large, the verification may fail. We use, therefore, iterative increment of the precision of floating point arithmetic. The following method is derived:

Algorithm 1 (Rational interpolation using computer assisted proof)

```
1. Set initial digits to the number $d$.
2. Solve the linear equations $Ay = B$ by Gaussian elimination and obtain an
   approximate solution $\tilde{y}$ and an approximate inverse matrix $R$ of $A$.
3. Verify the solution using Algorithm 3.1 in [7].
4. If nonsingularity of $A$ is verified, then output the result. Otherwise, increase
   the digits (for example, twice) and go to step 2.
```

This method should work well. In the example for $\log(x+2)$, starting from $d = 8$, the verification succeeds at precision $d = 16$. This gives us an accurate rational interpolant successfully as follows:

$$r_{4,4}(x) = 5.0787 \frac{(x + 19.851)(x + 4.8602)(x + 2.4752)(x + 1)}{(x + 35.346)(x + 7.0401)(x + 3.2192)(x + 2.1843)}.$$

TABLE 1. The error of rational approximations by HRFA

Interpolation $r_{m,n}(x)$	HRFA $\tilde{r}_{m,n}(x)$	Error E_{Ave}
$(7,7)$	$(6,6)$	$2.13 \cdot 10^{-6}$
$(8,8)$	$(4,4)$	$1.05 \cdot 10^{-5}$
$(9,9)$	$(6,6)$	$1.51 \cdot 10^{-4}$

TABLE 2. The error of rational interpolants by Algorithm 1

Interpolation $r_{m,n}(x)$	Precision d	Error E_{Ave}
$(7,7)$	32	$1.12 \cdot 10^{-24}$
$(8,8)$	32	$1.12 \cdot 10^{-28}$
$(9,9)$	64	$9.20 \cdot 10^{-33}$

Tables 1 and 2 show numerical comparisons between the error of the approximation by HRFA and Algorithm 1.

The function $f(x) = e^{x+2}$, $x \in [0,1]$ is approximated with the equidistant points from 0 to 1. The error of a rational approximation is estimated by the following expression

$$E_{\text{Ave}} := \frac{\sum_{i=1}^{100}(f(x_i) - r_{m,n}(x_i))/f(x_i)}{100},$$

where $x_i = (i-1)/99$.

The results by HRFA are shown in Table 1. Here, naïve rational interpolants are computed at precision $d = 8$, and we denote the degree of the rational function $r_{m,n}(x)$ by (m,n). For example, $r_{7,7}(x)$ has an undesired pole in the range $x \in [0,1]$. HRFA gives a reduced rational approximation $\tilde{r}_{6,6}(x)$ after eliminating the undesired pole, where the approximate GCD is computed by QRGCD in Maple 10 with parameter $\epsilon = 10^{-4}$. The results in Table 1 show that HRFA gives accurate approximations at the fixed precision.

The results by Algorithm 1 are shown in Table 2. For the computation of the rational interpolants $r_{7,7}(x)$, $r_{8,8}(x)$ and $r_{9,9}(x)$, starting at precision $d = 8$, the verification succeeds at precision $d = 32$, $d = 32$ and $d = 64$, respectively. Table 2 shows that, if multiple precision arithmetic is available, then we have accurate rational interpolants by Algorithm 1.

4. Conclusion

As already shown by M.-T. Noda and the author, hybrid rational function approximation (HRFA) works well for naïve rational function approximations. HRFA gives accurate approximation for given functions. One of the remaining problems is to consider a reason for the appearance of undesired zeros and poles. In this paper, we discuss the problem and show that it depends on the ill-conditioned property of the system of linear equations, which determines the coefficients of the rational interpolants. A straihgtforward method to eliminate the feature may be useful to give an accurate rational interpolant for a given data set.

References

[1] B. Beckermann, The condition number of real Vandermonde, Krylov and positive definite Hankel matrices, *Numer. Math.*, **85**, pp. 553–577, 2000.

[2] J. W. Demmel, On condition numbers and the distance to the nearest ill-posed problem, *Numer. Math.*, **51**, pp. 251–289, 1995.

[3] H. Kai and M.-T. Noda, Hybrid rational function approximation and its accuracy analysis, *Reliable Computing*, **6**, pp. 429–438, 2000.

[4] Y. Murakami, H. Kai and M.-T. Noda, Approximate algebraic computation for rational function approximations, *ACA '2002*, http://www.hpc.cs.ehime-u.ac.jp/˜aca2002/abstract/murakami.pdf, 2002.

[5] Y. Murakami, H. Kai and M.-T. Noda, Hybrid rational function approximation and ill-conditioned property, *J. JSSAC*, **11**(3,4), pp. 141–152, 2005 (in Japanese).

[6] M.-T. Noda, E. Miyahiro and H. Kai, Hybrid rational function approximation and its use in the hybrid integration, *Advances in Computer Methods for Partial Differential Equations VII*, edited by R. Vichnevetsky, D. Knight and G. Richter, pp. 565–571, IMACS, 1992.

[7] S. Oishi and S. M. Rump, Fast verification of solutions of matrix equations, *Numer. Math.*, **90**, pp. 755–773, 2002.

[8] R. Kacheong Yeung, Stable Rational Interpolation, Ph.D, Computing Science, University of Alberta, 1999.

Hiroshi Kai
Department of Computer Science
Ehime University
3 Bunkyo-cho, Matsuyama, Ehime 790-8977 Japan
e-mail: `kai@cs.ehime-u.ac.jp`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 55–68
© 2007 Birkhäuser Verlag Basel/Switzerland

Computing Approximate GCD of Multivariate Polynomials

Masaru Sanuki

Abstract. A new algorithm for computing the approximate GCD of multivariate polynomials is proposed by modifying the PC-PRS algorithm for exact GCD. We have implemented the new algorithm and compared it by typical examples with (approximate) PRS, (approximate) EZ-GCD algorithms and two new algorithms based on SVD. The experiment shows a good performance of our algorithm.

Keywords. Approximate GCD, polynomial remainder sequence (PRS), power-series coefficient PRS (PC-PRS) algorithm, extended Zassenhaus GCD (EZ-GCD) algorithm, singular value decomposition (SVD).

1. Introduction

Let $\mathbb{C}[x, u_1, \ldots, u_\ell]$ be the ring of polynomials over the complex number field $\mathbb{C}$ in the main variable x and sub-variables $u_1, \ldots, u_\ell$. We abbreviate $\mathbb{C}[x, u_1, \ldots, u_\ell]$ to $\mathbb{C}[x, u]$. Let a given polynomial $F(x, u) \in \mathbb{C}[x, u]$ be expressed as $F(x, u) = f_m(u)x^m + f_{m-1}(u)x^{m-1} + \cdots + f_0(u),\ f_m \neq 0$. By $\deg(F), \mathrm{lc}(F)$ and $\mathrm{tdeg}_u(f_i)$, we denote the degree and the leading coefficient of F w.r.t. the main variable x, and the total-degree of f_i w.r.t. $u_1, \ldots, u_\ell$, respectively; if $T = cu_1^{e_1} \cdots u_\ell^{e_\ell}, c \in \mathbb{C}$, then $\mathrm{tdeg}_u(T) = e_1 + \cdots + e_\ell$. By $\mathrm{tdeg}_u(F)$ and $\mathrm{tdeg}(F)$, we denote the total-degree of F w.r.t. $u_1, \ldots, u_\ell$ and w.r.t. $x, u_1, \ldots, u_\ell$, respectively, i.e., $\mathrm{tdeg}_u(F) = \max\{\mathrm{tdeg}_u(f_m), \ldots, \mathrm{tdeg}_u(f_0)\}$. By $||F||$, we denote the norm of polynomial F; we use the infinity norm in this paper: $||F|| = \max\{||f_m||, \ldots, ||f_0||\}$. By Normalize$(F)$, we denote the normalization of the polynomial F, i.e. the scale transformation $F \to F' = \eta F$, $\eta \in \mathbb{C}$, so that we have $||F'|| = 1$. By appGCD(A, B) and $\gcd(A, B; \varepsilon)$, we denote the approximate greatest common divisor (appGCD) of A and B and of tolerance ε, respectively. By $\mathrm{cont}(F; \varepsilon)$ and $\mathrm{pp}(F; \varepsilon)$, we denote the content of F of tolerance ε and the primitive part of F of tolerance ε, respectively, w.r.t. x, i.e., $\mathrm{cont}(F; \varepsilon) = \gcd\big(f_m, \gcd(f_{m-1}, \ldots, f_0; \varepsilon); \varepsilon\big)$ and $\mathrm{pp}(F; \varepsilon) = F/\mathrm{cont}(F; \varepsilon)$.

Zhi and Noda [ZN00] compared three algorithms for the appGCD of multivariate polynomials $A(x, u)$ and $B(x, u)$ experimentally, the polynomial remainder sequence (PRS) algorithm by Ochi *et al.* [ONS91], the EZ-GCD algorithm by Moses and Yun [MY73] with enhancement by Wang [Wan80], and a modular algorithm by Corless *et al.* [CGTW95]. They concluded from several experiments as follows. (a) The PRS algorithm is useful for the appGCD of polynomials of small degrees w.r.t. each variable. It is, however, quite inefficient when A and B are of large degrees. Furthermore, the coefficient size grows exponentially w.r.t. the number of variables. (b) EZ-GCD algorithm is fast, but it often causes large cancellation errors. Furthermore, if initial factors $A^{(0)}(x)$ and $B^{(0)}(x)$ of the Hensel construction have a close root, then multivariate Hensel construction becomes unstable [SY98]. (c) The Modular algorithm is often unstable.

At ISSAC 2004, two new methods for computing appGCD of multivariate polynomials were proposed. Gao *et al.* [GKMYZ04] proposed their appGCD algorithm as one part of an approximate bivariate factorization algorithm using Gao's factorization algorithm. Zeng and Dayton [ZD04] gave a nearly identical GCD algorithm that has an additional Gauss-Newton refinement step. In both papers, the authors proposed generalized Sylvester matrices and determined appGCDs by computing nearest singular matrices of the Sylvester matrices.

As for exact GCD, Sasaki and Suzuki [SS92] presented the so-called Power-series Coefficient PRS (PC-PRS) algorithm. The PC-PRS algorithm is a modification of the PRS algorithm, using truncated power series for the coefficients, and it is as fast as the EZ-GCD algorithm. In this paper, we modify the PC-PRS algorithm for calculating the appGCD and test it by several examples; the results are very good. We also compare the PC-PRS algorithm with the PRS, EZ-GCD and SVD-based algorithms for computing appGCD in Maple, MATLAB and GAL (see Section 4). The comparison shows a good performance of the PC-PRS algorithm. However, the PC-PRS algorithm becomes unstable in some case, and we discuss the case.

2. Approximate PC-PRS Algorithm

Let P_1 and P_2 be primitive polynomials in $\mathbb{C}[x, u]$ and $(P_1, P_2, P_3, \ldots, P_k \neq 0, P_{k+1} = 0)$ be a PRS. Let $P_k = C(u)G(x, u)$ where $C(u)$ is the content of P_k; then $G(x, u)$ is the GCD of P_1 and P_2. Usually, $\mathrm{tdeg}_u(G)$ is not large, but $\mathrm{tdeg}_u(C)$ is very large. If P_k and C are given, we can compute G by dividing P_k by C. In this division, we need not treat all the terms of P_k and C. For example, suppose $\mathrm{tdeg}_u(P_k) = 100$ and $\mathrm{tdeg}_u(C) = 90$; then $\mathrm{tdeg}_u(G) = 10$. In this case, the computation of G requires only terms of highest 10 exponents or terms of lowest 10 exponents of P_k and C. In the PC-PRS algorithm, we use only terms of lowest some exponents, and cut-off all the other higher terms, which makes the computation quite fast.

Let e be an upper bound of $\mathrm{tdeg}_u(G)$; we discard all the terms of total-degrees greater than e. In order to cut-off terms simply, we introduce the total-degree variable t for the sub-variables $u_1, \ldots, u_\ell$ by transformation $u_i \mapsto tu_i$ $(i = 1, \ldots, \ell)$. Let $\tilde{P}_1 = P_1(x, tu), \tilde{P}_2 = P_2(x, tu) \in \mathbb{C}\{tu\}[x]$, where $\mathbb{C}\{tu\}$ is the power-series ring over $\mathbb{C}$. We compute a PRS with power-series coefficients, or PC-PRS, $(\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_k \not\equiv 0, \tilde{P}_{k+1} \equiv 0)$ by truncating terms of total-degree greater than e; hence $\tilde{P}_i \equiv P_i \pmod{t^{e+1}}$ $(i = 1, 2, \ldots)$. Actually, $\tilde{P}_i$ is generated as follows:

$$\begin{cases} \beta_i \tilde{P}_{i+1} \equiv (\alpha_i \tilde{P}_{i-1} - Q_i \tilde{P}_i)/\max\{||\alpha_i||, ||Q_i||\} \pmod{t^{e+1}}, \\ \alpha_i = \mathrm{lc}(\tilde{P}_i)^{d_i+1}, \quad d_i = \deg(\tilde{P}_{i-1}) - \deg(\tilde{P}_i), \end{cases}$$

where β_i is determined by the reduced-PRS or the subresultant PRS algorithm. By modifying the PC-PRS algorithm, we obtain the following approximate PC-PRS algorithm. (The algorithm does not consider the case of small leading coefficients. The treatment of PRS with small leading coefficients is now an open problem.)

Algorithm 1 (approximate PC-PRS algorithm).

```
 Input:   Polynomials $P_1, P_2 \in \mathbb{C}[x, u]$ and a small number $\varepsilon$.
Output:   $G = \gcd(P_1, P_2; \varepsilon)$.
STEP 1:   $g := \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2); \varepsilon)$.
          $E := \min_{i=1,2}\{\mathrm{tdeg}_u(P_i) + \mathrm{tdeg}_u(g) - \mathrm{tdeg}_u(\mathrm{lc}(P_i))\}$.
STEP 2:   Calculate PRS $(\tilde{P}_3, \ldots, \tilde{P}_k, \tilde{P}_{k+1}, ||\tilde{P}_{k+1}||/||\tilde{P}_k|| \le \varepsilon)$.
             /* cut-off higher degree $E$ terms */
          if $\deg(\tilde{P}_k) = 0$ then return 1 else $\tilde{P}_k :=$ Normalize$(\tilde{P}_k)$.
          $\tilde{P} := g\tilde{P}_k/\mathrm{lc}(\tilde{P}_k)$.   /* power-series division */
          $G := \mathrm{pp}(\tilde{P}; \varepsilon)$.
          if $||\mathrm{rem}(P_1, G)|| \le \varepsilon$ and $||\mathrm{rem}(P_2, G)|| \le \varepsilon$
             then return $G$ else return 1.
          end.
```

3. Other Algorithms

3.1. EZ-GCD Algorithm

The EZ-GCD algorithm may be described as follows.

1. Choose a lucky expansion point $s \in \mathbb{C}^\ell$ and put $I = (u_1 - s_1, \ldots, u_\ell - s_\ell)$.
2. Put $A^{(0)} \equiv A,\ B^{(0)} \equiv B \pmod{I}$, and compute $G^{(0)} = \gcd(A^{(0)}, B^{(0)})$.
3. Find a and $b \in \mathbb{C}$ such that $\gcd(G^{(0)}, H^{(0)}) = 1$ where $P^{(0)} = aA^{(0)} + bB^{(0)}$ and $H^{(0)} = P^{(0)}/G^{(0)}$, and put $P = aA + bB$.
4. Compute polynomials $U_i(x)$ and $V_i(x)$ $(i = 0, 1, \ldots, \deg(P^{(0)}))$ such that
$$U_i(x)G^{(0)} + V_i(x)H^{(0)} = x^i.$$
5. Perform the Hensel construction to obtain polynomials $G^{(k)}$ and $H^{(k)}$:
$$P(x, u) \equiv G^{(k)}(x, u)H^{(k)}(x, u) \pmod{I^{k+1}},$$

where ($n = \deg(P^{(0)})$ below)

$$G^{(k)} = G^{(k-1)} + \sum_{i=0}^{n} V_i f_i^{(k)}, \quad H^{(k)} = H^{(k-1)} + \sum_{i=0}^{n} U_i f_i^{(k)},$$

with

$$F^{(k)} \equiv P - G^{(k-1)}H^{(k-1)} = f_n^{(k)}x^n + \cdots + f_0^{(k)} \pmod{I^{k+1}}.$$

We compute $U_i(x)$ and $V_i(x)$ by the extended Euclidean algorithm.

6. If $G^{(k)}|A$ and $G^{(k)}|B$ then return $G^{(k)}$, else choose another expansion point $s' \in \mathbb{C}^\ell$ and go to step 1.

3.2. SVD-Based Algorithms

The algorithms of Gao *et al.* [GKMYZ04] and Zeng-Dayton [ZD04] for computing appGCD of multivariate polynomials are based on singular value decomposition (SVD) and are very similar to each other. Let $f, g \in \mathbb{C}[x_1, \ldots, x_\ell]$, $f_1 = f/\gcd(f,g)$ and $g_1 = g/\gcd(f,g)$. Then all the solutions $u, v \in \mathbb{C}[x_1, \ldots, x_\ell]$ of the equation

$$uf + vg = 0 \tag{3.1}$$

are of the form

$$u = g_1 q, \quad v = -f_1 q, \tag{3.2}$$

where $q \in \mathbb{C}[x_1, \ldots, x_\ell]$ (Lemma 2.1 in [GKMYZ04]). From the equation (3.1), we can derive a linear system for the coefficients of u and v, and we can obtain solutions $u = g_1$ and $v = -f_1$ by solving the linear system.

Let S_k and $S_{\mathbf{k}}$ be Gao *et al.*'s k-th generalized Sylvester matrix and Zeng-Dayton's $\mathbf{k}$-th generalized Sylvester matrix, respectively. Then $\mathrm{tdeg}(\gcd(f,g)) = k$ (or $\mathrm{ldeg}(\gcd(f,g)) = \mathbf{k}$ (for ldeg, see Sect. 3.2.2)) if and only if S_k (or $S_{\mathbf{k}}$) is rank-deficient by one with its nullspace being spanned by $[\mathbf{v}, -\mathbf{u}]^T$, where $\mathbf{u}$ and $\mathbf{v}$ are coefficient vectors corresponding to u and v, respectively. Then, the condition of degree of GCD tells us that $u = g_1$ and $v = -f_1$. Gao *et al.*'s and Zeng-Dayton's algorithms are summarized as follows.

1. Determine k (or $\mathbf{k}$), the total-degree (or the ℓ-degree) w.r.t. $x_1, \ldots, x_\ell$ of the appGCD(f,g) in one of the following two ways:
   a) Computing the degrees of the GCDs of several random univariate projections of f and g, and look for the numerical rank of the corresponding univariate Sylvester matrices.
   b) (Gao *et al.*'s algorithm only) From $S = S_1(f,g)$ where $f, g \in \mathbb{C}[x_1, \ldots, x_\ell]$ with $\mathrm{tdeg}(u) < \mathrm{tdeg}(g)$ and $\mathrm{tdeg}(v) < \mathrm{tdeg}(f)$, find the largest gap in the singular values of S and infer the degree from the numerical rank of S.
2. For the k-th (or $\mathbf{k}$-th) generalized Sylvester matrix S_k (or $S_{\mathbf{k}}$), compute a basis of the nullspace by computing the singular vector corresponding to the smallest singular value of S_k (or $S_{\mathbf{k}}$).

3. Find a $G = \gcd(f, g)$, the approximate quotient of f and v (or g and u); alternatively minimize $||g - Gu||_2^2 + ||f + Gv||_2^2$, by using a least-square algorithm.

We note that the generalized Sylvester matrices are different in Gao *et al.*'s and Zeng-Dayton's papers. The size of S_k is smaller than the size of $S_{\mathbf{k}}$ (see Example 5). Also, we note that Gao *et al.*'s algorithm does not require tolerance ε, but Zeng-Dayton's algorithm requires tolerance ε because of determining degree of univariate polynomial GCDs.

3.2.1. Gao et al.'s k-th Generalized Sylvester Matrix. Let $\mathrm{tdeg}(f) = m$, $\mathrm{tdeg}(g) = n$, and $\beta(k, \ell) = \binom{k+\ell}{\ell}$; number of all the different terms $x_1^{i_1} \cdots x_\ell^{i_\ell}$ with $i_1 + \cdots + i_\ell \le k$. Then, Gao *et al.*'s k-th Sylvester matrix

$$S_k \in \mathbb{C}^{\beta(m+n-k,\ell) \times (\beta(m-k,\ell)+\beta(n-k,\ell))}. \tag{3.3}$$

3.2.2. Zeng-Dayton's k-th Generalized Sylvester Matrix. For $f \in \mathbb{C}[x_1, \ldots, x_\ell]$, let $m_i = \deg_{x_i}(f)$ $(i = 1, \ldots, \ell)$. By $\mathrm{ldeg}(f)$, or ℓ–degree of f, we denote the ℓ–tuple $\mathbf{m} = [m_1, \ldots, m_\ell]$. By $\nu(\mathbf{m})$, we denote the number of all the different terms $x_1^{i_1} \cdots x_\ell^{i_\ell}$ with $i_j \le m_j$ $(1 \le j \le \ell)$ i.e., $\nu(\mathbf{m}) = (m_1 + 1)(m_2 + 1) \cdots (m_\ell + 1)$. In Zeng-Dayton's algorithm, every polynomial is transformed to a coefficient vector; if $f = \sum_{i_1 + \cdots i_\ell \le \nu(\mathbf{m}),\ i_j \le m_j} f_s x_1^{i_1} \cdots x_\ell^{i_\ell}$ with $s = 1 + \sum_{k=1}^{\ell} \left[i_k \prod_{j=1}^{k-1} (m_j + 1) \right]$, then

$$\Psi_{\mathbf{m}} : f \mapsto \mathbf{f} = [f_1, f_2, \ldots, f_{\nu(\mathbf{m})}]^T \in \mathbb{C}^{\nu(m)}.$$

Let $\mathbf{n} = [n_1, \ldots, n_\ell]$. By $C_{\mathbf{n}}(f) \in \mathbb{C}^{\nu(\mathbf{m}+\mathbf{n}) \times \nu(\mathbf{m})}$, we denote a convolution matrix of f; the s-th column of $C_{\mathbf{n}}(f)$ is generated by $\Psi_{\mathbf{m}+\mathbf{n}}(f q_s)$, where $q_s = x_1^{i_1} \cdots x_\ell^{i_\ell}$ with $s = 1 + \sum_{k=1}^{\ell} \left[i_k \prod_{j=1}^{k-1} (n_j + 1) \right]$ $(1 \le s \le \nu(\mathbf{m}))$. Let $\mathrm{ldeg}(g) = \mathbf{n}$; then Zeng-Dayton's $\mathbf{k}$-th Sylvester matrix

$$S_{\mathbf{k}}(f, g) = [C_{\mathbf{m}-\mathbf{k}}(g) | C_{\mathbf{n}-\mathbf{k}}(f)] \in \mathbb{C}^{\nu(\mathbf{m}+\mathbf{n}-\mathbf{k}) \times (\nu(\mathbf{m}-\mathbf{k})+\nu(\mathbf{m}-\mathbf{k}))}. \tag{3.4}$$

4. Numerical Examples

We have implemented the Approximate PC-PRS algorithm in Maple and GAL, and Zeng-Dayton's algorithm in MATLAB. GAL is an algebra system constructed by Sasaki and Kako, being equipped with *effective floating-point number* system which allows us to detect the cancellation errors fairly well but not exactly; see [KS97] for details (since the relative errors are set to 10^{-15} initially, the predicted values are about 10 times larger than the actual values). We use GAL to estimate cancellation errors of PRS, EZ-GCD and PC-PRS, and we use Maple and MATLAB to compare average CPU times for the PC-PRS and SVD-based algorithms. We have used Gao's code "multigcd" is Maple for Gao *et al.*'s algorithm [Kal04, Zhi04]. We compare our algorithm with Gao *et al.*'s algorithm in Maple and Zeng-Dayton's algorithm in MATLAB. Since the size of generalized Sylvester matrices is very large in most cases, we have implemented `sparse` in

MATLAB [Matlab]. Computations reported in this paper were performed on an Xeon (and Ultra SPARC-IIi) running at 3.05GHz (and 440MHz) under Windows XP (and Solaris 8), using Maple 9.5, MATLAB 7.1 (and GAL).

An Execution of Approximate PC-PRS Algorithm

Example 1. Let multivariate polynomials $A(x, y, z)$ and $B(x, y, z)$ be

$$A(x, y, z) = (x^2 + y^2 + 0.3z^3 - 1)^2(xy - 0.25) - 10^{-5}xyz,$$
$$B(x, y, z) = (x^2 + y^2 + 0.3z^3 - 1)(x - y)^3 - 10^{-5}(x + 1 - z).$$

The approximate PC-PRS algorithm, with $\varepsilon = 0.001$, works as follows.

- $g = \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2); \varepsilon) = \gcd(y, 1; \varepsilon) = 1$.
- $E = 6$.
- Calculate PC-PRS:
  $\deg(\tilde{P}_1), \ldots, \deg(\tilde{P}_6)$ are $5, 5, 4, 3, 2, 1$, respectively.
  $||\tilde{P}_1||, \ldots, ||\tilde{P}_5||$ are $O(1)$ and $||\tilde{P}_6|| \le O(\varepsilon)$.
- Normalization of P_5:
  $\tilde{P}_5 = \mathrm{Normalize(P_5)} = 0.8799\cdots x^2y^4 + 0.0959\cdots x^2y^2z^3 - 0.1199\cdots x^2y^2 -0.0060\cdots x^2z^3 + 0.0066\cdots x^2 - y^4 - 0.1380\cdots y^2z^3 + 0.1266\cdots y^2 +0.0080\cdots z^3 - 0.0066\cdots$.
- Power-series division:
  $\tilde{P} = g\tilde{P}_5/\mathrm{lc}(\tilde{P}_5) = x^2 + 0.9999\cdots y^2 + 0.2999\cdots z^3 - 1.0000\cdots$.
- $G = \mathrm{pp}(\tilde{P}; \varepsilon) = \tilde{P}$.
- Check $||\mathrm{rem}(P_1, G)|| \le \varepsilon$ and $||\mathrm{rem}(P_2, G)|| \le \varepsilon$.
  $||\mathrm{rem}(P_1, G)|| = 0.0001 < 0.001$, $||\mathrm{rem}(P_2, G)|| = 0.00019 < 0.001$.
- Return G.

Example 2 (Comparison of PRS algorithm with PC-PRS algorithm). We compare the PRS and PC-PRS algorithms on the following $A(x, y, z)$ and $B(x, y, z)$:

$$A(x, y, z) = (x^2 + y^{2m} + 0.3z^{3m} - 1)^2(xy - 0.25) - 10^{-5}xyz^n,$$
$$B(x, y, z) = (x^2 + y^{2m} + 0.3z^{3m} - 1)(x - y^n)^3 - 10^{-5}(x + 1 - z).$$

Cancellation Errors by PRS and PC-PRS

In Tables 1 and 2, "#term" and "ErrMax" show numbers of terms and maximum relative errors in the coefficients $P_i (i = 1, \ldots, 6)$, respectively. "–Error–" means that the computation stopped by failure (in this case, error happened in the approximate division $\tilde{P}/\mathrm{cont}(\tilde{P}; \varepsilon)$ with tolerance ε).

Comparison of Computing Times

- Table 3: Average CPU times for $m = 1$ and $n = 1, \ldots, 6$.
  The total-degree of appGCD is the same for different n.
- Table 4: Average CPU times for $n = 1$ and $m = 1, \ldots, 4$.
  The total-degree of appGCD increases as m increases.

TABLE 1. Approximate PRS Algorithm

	#term	ErrMax(P_i)
P_1	21	1.00e-15
P_2	17	1.00e-15
P_3	27	3.00e-15
P_4	58	2.70e-14
P_5	166	1.20e-7
P_6		$O(0.0001)$
approx. GCD		— Error —

TABLE 2. Approximate PC-PRS Algorithm

	#term	ErrMax(P_i)
P_1	21	1.00e-15
P_2	17	1.00e-15
P_3	15	2.00e-15
P_4	19	8.00e-15
P_5	22	1.62e-9
P_6		$O(0.0001)$
approx. GCD		2.53e-13

TABLE 3. The average CPU times for $m = 1$ and $n = 1, \dots, 6$ (sec)

$m = 1$	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
Approx. PRS	0.202	0.828	0.424	1.408	1.380	1.714
Approx. PC-PRS	0.016	0.016	0.012	0.016	0.014	0.012

TABLE 4. The average CPU times for $n = 1$ and $m = 1, \dots, 4$ (sec)

$n = 1$	$m = 1$	$m = 2$	$m = 3$	$m = 4$
Approx. PRS	0.202	0.370	0.694	0.744
Approx. PC-PRS	0.016	0.026	0.036	0.058

We see that the approximate PC-PRS algorithm shows a very nice efficiency. The average CPU time for approximate PRS algorithm depends on total-degrees of A and B largely. On the other hand, the average CPU time for the approximate PC-PRS algorithm depends only on the total-degree of appGCD, and they do not depend much on other quantities.

Comparison of PC-PRS Algorithm with EZ-GCD Algorithm

Example 3. We compare the PC-PRS and EZ-GCD algorithms on the following $A(x, u)$ and $B(x, u)$:

$$\begin{cases} A(x,u) = (x^3 + u^4 + u^3 - u^2 + u + 1 + 0.001)(x^2 + xu + u^2 + 1), \\ B(x,u) = (x^3 + u^4 + u^3 - u^2 + u + 1 + 0.001)(x^2 + xu + 1). \end{cases}$$

1. Approximate PC-PRS Algorithm
   We obtain an appGCD(A, B) as follows:

$$0.999000999001x^3 + 0.999000999001u^4 + 0.999000999001u^3$$
$$-0.999000999001u^2 + 0.999000999001u + 1.$$

The maximum relative error of this appGCD is 2.00×10^{-15}.

2. EZ-GCD Algorithm
   We choose the expansion point at $s = 4/2005$ and put $a = 1$ and $b = 1$; then

$$G^{(0)} = 0.99701787928153x^3 + 1,$$
$$H^{(0)} = 2.0059820807004x^2 + 0.0040019592632428x + 2.0059860726797.$$

   $G^{(0)}$ and $H^{(0)}$ have mutually close roots, and the maximum relative errors of $G^{(0)}$ and $H^{(0)}$ are 2.51×10^{-10} and 2.51×10^{-10}, respectively. We obtain the following appGCD(A, B):

$$\begin{aligned}\tilde{G} = {} & 0.9970\cdots x^3 + 0.0181\cdots x^2u^4 - 0.0026\cdots x^2u^3 \\ & -0.0015\cdots x^2u^2 + 0.0019\cdots x^2u + 0.0177\cdots xu^4 - 0.0105\cdots xu^3 \\ & +0.0044\cdots xu^2 - 0.0019\cdots xu - 0.0306\cdots u^4 + 0.9942\cdots u^3 \\ & -0.9935\cdots u^2 + 0.9950\cdots u + 1.\end{aligned}$$

   The maximum relative error of this appGCD by the EZ-GCD algorithm is 6.33×10^{-6}. We find that remainders of A and B by $\tilde{G}$ are $O(1)$.

   Since we have chosen the expansion point s to be near a singular point (in this case, the origin is a singular point), the EZ-GCD algorithm causes large cancellation errors. We see that the approximate PC-PRS algorithm does not cause such a large cancellation error; it is independent of the singular point.

Example 4. We compare the PC-PRS and EZ-GCD algorithms on the following $A(x, u)$, $B(x, u)$ and $\tilde{B}(x, u)$:

$$\begin{cases} A(x, u) = (x^2 + u^2 + 1)(x^2 - u - 0.5)(x^2 + u + 0.1), \\ B(x, u) = (x^2 + u^2 + 1)(x + u^3 + u - 0.4)(\underline{0.001}x^2 + u + 1), \\ \tilde{B}(x, u) = (x^2 + u^2 + 1)(x + u^3 + u - 0.4)(\underline{0.0001}x^2 + u + 1). \end{cases}$$

1. Approximate PC-PRS Algorithm
   We obtain an appGCD(A, B) as follows

$$x^2 + u^2 + 1.$$

   The maximum relative error of this appGCD is 1.13×10^{-11}. However, we cannot obtain any appGCD$(A, \tilde{B})$: we can calculate the PRS of A and B as $(P_1 = A, P_2 = B, \ldots, P_5, P_6,\ ||P_6|| = O(0.001))$, but the PRS of A and $\tilde{B}$ is $(P_1 = A, P_2 = B, \ldots, P_5,\ ||P_5|| = O(0.001))$. Since $||\mathrm{lc}(B)||$ is smaller than the norms of other coefficients of B, the approximate PC-PRS algorithm causes large cancellation errors.
2. EZ-GCD Algorithm
   We choose the expansion point at $s = 0$ and put $a = 1$ and $b = 1$; then

$$G^{(0)} = x^2 + 0.99999999999811,$$
$$H^{(0)} = x^4 + 0.001x^3 - 0.40039999999811x^2 + 1.0x - 0.45000000000264.$$

   We obtain appGCD(A, B) and appGCD$(A, \tilde{B})$ as follows

$$x^2 + u^2 + 1.$$

The maximum relative errors of appGCD(A,B) and appGCD$(A,\tilde{B})$ by EZ-GCD algorithm are 1.13×10^{-11} and 1.13×10^{-11}, respectively.

We see that the approximate PC-PRS algorithm shows a very good efficiency. We also see that, in some cases, the EZ-GCD algorithm gives better results than the approximate PC-PRS algorithm.

Example 5 (Comparison of PC-PRS with SVD-based algorithms). In Tables 5, 6 and 7, "*back_err*" denotes the backward error in Zeng-Dayton's algorithm [ZD04] defined as follows:

$$back\_err = \sqrt{\frac{||\tilde{c}\tilde{f}_1 - f||_2^2 + ||\tilde{c}\tilde{g}_1 - g||_2^2}{||f||_2^2 + ||g||_2^2}}\ , \tag{4.1}$$

where $\tilde{f}_1, \tilde{g}_1$ and $\tilde{c}$ are co-factors of f, g and an appGCD(f,g), see below.

We compare the PC-PRS and SVD-based algorithms on the following f and g.

Sample 1.

$$\begin{cases} f(x,u) = f_1(x,u)c(x,u), \\ g(x,u) = g_1(x,u)c(x,u), \end{cases} \text{with} \begin{cases} c(x,u) = (x+u_1+\cdots+u_r+1)^2, \\ f_1(x,u) = (x^2-u_1-\cdots-u_r-0.5)^2, \\ g_1(x,u) = (x^2+u_1+\cdots+u_r+0.5)^2. \end{cases}$$

We set r as $r=1,2,\ldots,5$. We show the comparison in Table 5, where "Ave. CPU", "ErrMax" and "*back_err*" are the average CPU time (sec), the maximum relative error and the backward error, respectively.

TABLE 5. Comparison of PC-PRS with SVD-based GCDs on Sample 1

	Approx. PC-PRS (Maple)		Gao *et al.*'s GCD (Maple)		Zeng-Dayton's GCD (MATLAB)	
	Ave. CPU	ErrMax	Ave. CPU	*back_err*	Ave. CPU	*back_err*
$r=1$	0.015	5.12e-13	2.647	5.62e-15	0.072	5.80e-15
$r=2$	0.025	5.12e-13	10.286	1.16e-14	0.119	8.41e-14
$r=3$	0.052	5.12e-13	96.206	1.36e-10	0.756	1.84e-13
$r=4$	0.151	5.12e-13	about 1300	1.72e-10	1.519	1.16e-13
$r=5$	0.220	5.12e-13	over 3000	- - - -	446.755	3.94e-13

Sample 2.

$$\begin{cases} f(x,u) = f_2(x,u)c(x,u), \\ g(x,u) = g_2(x,u)c(x,u), \end{cases} \text{with} \begin{cases} c(x,u) = (x+u_1+\cdots+u_r^r+1)^2, \\ f_2(x,u) = (x^2-u_1-\cdots-u_r-0.5)^2, \\ g_2(x,u) = (x^2+u_1+\cdots+u_r+0.5)^2. \end{cases}$$

We set r as $r=1,2,\ldots,5$.

Table 6. Comparison of PC-PRS with SVD-based GCDs on Sample 2

	Approx. PC-PRS		Gao *et al.*'s GCD		Zeng-Dayton's GCD	
	Ave. CPU	ErrMax	Ave. CPU	*back_err*	Ave. CPU	*back_err*
$r = 1$	0.015	5.12e-13	2.647	5.62e-15	0.072	5.80e-15
$r = 2$	0.062	4.60e-12	11.114	1.30e-14	0.137	6.06e-14
$r = 3$	0.130	8.19e-12	247.329	1.91e-10	1.988	7.94e-13
$r = 4$	0.399	8.19e-12	over 3000	- - - -	338.754	7.88e-13
$r = 5$	3.736	8.19e-12	over 3000	- - - -	over 3000	- - - -

Sample 3.

$$\begin{cases} f(x,u) = f_3(x,u)c(x,u), \\ g(x,u) = g_3(x,u)c(x,u), \end{cases} \text{with} \begin{cases} c(x,u) = (x + u_1 + \cdots + u_r + 1)^2, \\ f_3(x,u) = (x^2 - u_1 - \cdots - u_r^r - 0.5)^2, \\ g_3(x,u) = (x^2 + u_1 + \cdots + u_r + 0.5)^2. \end{cases}$$

We set r as $r = 1, 2, \ldots, 5$.

Table 7. Comparison of PC-PRS with SVD-based GCDs on Sample 3

	Approx. PC-PRS		Gao *et al.*'s GCD		Zeng-Dayton's GCD	
	Ave. CPU	ErrMax	Ave. CPU	*back_err*	Ave. CPU	*back_err*
$r = 1$	0.015	5.12e-13	2.647	5.62e-15	0.072	5.80e-15
$r = 2$	0.082	5.12e-13	7.899	1.47e-14	0.163	4.87e-15
$r = 3$	0.151	5.12e-13	426.549	1.32e-10	83.323	2.62e-13
$r = 4$	0.438	5.12e-13	3000 over	- - - -	over 3000	- - - -
$r = 5$	2.167	5.12e-13	3000 over	- - - -	over 3000	- - - -

We see that the approximate PC-PRS algorithm shows a very nice efficiency. The CPU time for Gao *et al.*'s algorithm depends on the total-degrees of f and g. Furthermore, Gao *et al.*'s algorithm is extremely slowed down if the number of variables is large. On the other hand, Zeng-Dayton's algorithm is not so slowed down even if the number of variables is increased (Example 1 in [ZD04]). This difference between Gao *et al.*'s algorithm and Zeng-Dayton's algorithm is due to computing environments: efficiency of Zeng-Dayton's algorithm is strongly due to very fast sparse matrix SVD procedures in MATLAB. If we implement Gao *et al.*'s algorithm in MATLAB, the algorithm will show a much better performance. Observing the maximum relative error and the backward error, the approximate PC-PRS algorithm is stable if the total-degree and the number of sub-variables are increased. On the other hand, SVD-based algorithms lose stability gradually.

Comparison of S_k with $S_{\mathbf{k}}$.

The size of the generalized Sylvester matrix increases very rapidly as the degree or the number of variables increases. The generalized Sylvester matrices S_k and

$S_{\mathbf{k}}$ are different. Table 8 shows the sizes of generalized Sylvester matrices S_k and $S_{\mathbf{k}}$ for Samples 1 and 3 above.

TABLE 8. Sizes of generalized Sylvester matrices

	Sample 1 (Table 5)		Sample 3 (Table 7)	
	S_k	$S_{\mathbf{k}}$	S_k	$S_{\mathbf{k}}$
$r=1$	66×30	77×30	66×30	77×30
$r=2$	286×70	539×90	286×70	819×120
$r=3$	1001×140	3773×270	1829×280	9009×660
$r=4$	3003×252	26411×810	20349×3129	99099×5130
$r=5$	8008×420	184877 ×2430	74613 ×8218	1756755 ×53190

We see that size$(S_k) \le$ size$(S_{\mathbf{k}})$. By row(M) and col(M), we denote the numbers of rows and columns, respectively, of a matrix M. For Gao *et al.*'s Sylvester matrix S_k, row$(S_k) = \beta(m+n-k, \ell) = \binom{m+n-k+\ell}{\ell}$ and col$(S_k) = \beta(m-k, \ell) + \beta(n-k, \ell) = \binom{m-k+\ell}{\ell} + \binom{n-k+\ell}{\ell}$. For Zeng-Dayton's Sylvester matrix $S_{\mathbf{k}}$, row$(S_{\mathbf{k}}) = \prod_{i=1}^{\ell}(m_i+n_i-k_i+1)$ and col$(S_{\mathbf{k}}) = \prod_{i=1}^{\ell}(m_i-k_i+1)+\prod_{i=1}^{\ell}(n_i-k_i+1)$. We have

$$\begin{aligned}\binom{m-k+\ell}{\ell} &= \frac{1}{\ell!}\prod_{i=1}^{\ell}(m-k+\ell-i+1)\\ &\le \frac{1}{\ell!}\Big\{(m-k+1)^{\ell} + \ell^2(m-k+1)^{\ell-1}\\ &\qquad +\ell^3(m-k+1)^{\ell-2} + \cdots + \ell^{\ell}\Big\}\\ &\le \frac{\ell+1}{\ell!} \times \Big(\max\{m-k+1, \ell\}\Big)^{\ell}, \qquad (4.2)\end{aligned}$$

$$\prod_{i=1}^{\ell}(m_i-k_i+1) \ge \prod_{i=1}^{\ell}(\breve{m}-\breve{k}+1) = (\breve{m}-\breve{k}+1)^{\ell}, \qquad (4.3)$$

where $(\breve{m}, \breve{k})$ is a pair which minimizes $m_i - k_i$ $(i = 1, \ldots, \ell)$. Since $m-k \ge \breve{m}-\breve{k}$, we can bound row$(S_k)$ and col(S_k) as follows:

$$\mathrm{row}(S_k) \le \frac{\ell+1}{\ell!}\Big(\max\{m+n-k+1, \ell\}\Big)^{\ell} \le (\breve{m}+\breve{n}-\breve{k}+1)^{\ell} \le \mathrm{row}(S_{\mathbf{k}}),$$

$$\begin{aligned}\mathrm{col}(S_k) &\le \frac{\ell+1}{\ell!}\Big(\max\{m-k+1, \ell\} + \max\{n-k+1, \ell\}\Big)^{\ell}\\ &\le (\breve{m}-\breve{k}+1) + (\breve{n}-\breve{k}+1) \le \mathrm{col}(S_{\mathbf{k}}).\end{aligned}$$

Therefore, size$(S_{\mathbf{k}})$ is larger than size(S_k). In most cases, $\max\{m+n-k+1, \ell\} \approx \breve{m}+\breve{n}-\breve{k}+1$; hence we have

$$\mathrm{row}(S_k) \lesssim \Big(\frac{\ell+1}{\ell!}\Big)\mathrm{row}(S_{\mathbf{k}}). \qquad (4.4)$$

Example 6. We compare the performance of approximate PC-PRS and Gao *et al.*'s GCD by multivariate polynomials generated randomly in Maple. We generate ten polynomial pairs (A_i, B_i) $(i = 1, \ldots, 10)$ as follows:

$$\begin{cases} A_i = a_i c_i + 10^{-2} d_i \\ B_i = b_i c_i + 10^{-2} e_i \end{cases} \quad (i = 1, \ldots, 5),$$

$$\begin{cases} A_{i+5} = a_i c_i + 10^{-5} d_i \\ B_{i+5} = b_i c_i + 10^{-5} e_i \end{cases} \quad (i = 1, \ldots, 5),$$

where $a_i, b_i, c_i, d_i, e_i \in \mathbb{C}[x, y]$ $(i = 1, 2, 3)$ and $a_i, b_i, c_i, d_i, e_i \in \mathbb{C}[x, y, z]$ $(i = 4, 5)$, with $||a_i|| = ||b_i|| = ||c_i|| = ||d_i|| = ||e_i|| = 1$ $(i = 1, \ldots, 5)$, and we generate them randomly in Maple. Table 9 shows the comparison.

Table 9. Comparison of PC-PRS with SVD-Based GCDs (sec)

	Approx. PC-PRS (Maple)		Gao *et al.*'s GCD (Maple)		Zeng-Dayton's GCD (MATLAB)	
Ex.	Ave. CPU	ErrMax	Ave. CPU	*back_err*	Ave. CPU	*back_err*
1	0.025	7.27e-12	0.871	9.65e-3	0.094	3.52e-1
2	0.044	3.00e-14	1.825	1.66e-2	0.094	4.83e-1
3	0.029	1.37e-13	3.670	1.25e-2	0.090	5.13e-1
4	0.110	3.47e-7	4.892	1.66e-2	0.253	3.32e-0
5	— Error —		4.119	1.18e-2	0.294	2.44e-0
6	0.037	4.68e-11	2.170	9.65e-6	0.098	2.08e-5
7	0.040	1.39e-13	1.121	1.66e-5	0.087	1.80e-5
8	0.039	1.66e-9	0.341	1.25e-5	0.082	1.95e-5
9	0.049	3.22e-6	2.503	1.66e-5	0.102	2.24e-5
10	0.471	1.90e-8	2.563	1.18e-5	0.100	1.76e-5

We see from Table 9 that Gao *et al.*'s algorithm is unstable if the perturbation is not small; it is difficult to determine the total-degree of appGCD by only the SVDs of univariate polynomials. For example, in Ex. 3, we obtain an appGCD of the total-degree 1, but $\mathrm{tdeg}(\gcd(A_3, B_3)) = 3$. In Gao *et al.*'s algorithm, we determine the total-degree by rank deficiency by observing the largest gap of singular values, and hence the determination is unstable if perturbation is not small. The situation is the same in Zeng-Dayton's algorithm. Furthermore, since Zeng-Dayton's algorithm requires degrees of ℓ univariate polynomial GCDs for ℓ variables, it is more unstable if the number of variables is large. In fact, Zeng-Dayton's algorithm does not give correct appGCD for Ex. 1–5. In Ex. 4 and 8, $||\mathrm{lc}(B_4)||$ and $||\mathrm{lc}(B_9)||$ are small compared with the norms of other coefficients. Therefore, the approximate PC-PRS algorithm causes large cancellation errors. On the other hand, Gao *et al.*'s algorithm and Zeng-Dayton's algorithm are not affected by norms of coefficients. In Ex. 5, "Error" means that the computation stopped by failure (in this case, an error happened in computing an appGCD of $\mathrm{lc}(A_5)$ and $\mathrm{lc}(B_5)$; $\gcd(\mathrm{lc}(A_5), \mathrm{lc}(B_5); 0.01) = \gcd(0, \mathrm{lc}(B_5); 0.01) = 0$. The reason is the small leading coefficient: $||\mathrm{lc}(A_5)|| < 0.01$). The approximate PC-PRS algorithm is not complete yet. Improvement of the algorithm is our future work.

5. Conclusion

In this paper, we proposed an Approximate PC-PRS algorithm and briefly discuss five algorithms for computing appGCD of multivariate polynomials. Our algorithm cut-offs unnecessary higher degree terms in the computation of PRS, making the computation quite efficient. We showed a good performance of the approximate PC-PRS algorithm. This algorithm is very nice except in the case of a small leading coefficient (Example 4 and Ex. 5 in Example 6). SVD-based algorithms can compute appGCD fast, only when the degree and number of variable are small. The size of the generalized Sylvester matrix increases very rapidly as the degree and the number of variables are increased; hence SVD-based algorithms require very fast SVD routines such as those in MATLAB. SVD-based algorithms are unstable when the perturbation is not small; we cannot determine the degree of appGCD. On the other hand, SVD-based algorithms are not affected by norms of coefficients.

Acknowledgments. The author expresses his sincere thank to Prof. Tateaki Sasaki, his master thesis supervisor, who gave him a theme of modifying the PC-PRS GCD algorithm to an approximate one and helped him variously. He also thanks Prof. Zhonggang Zeng for presenting him the multivariate GCD package MVGCD in MATLAB code [ZD04].

References

[CGTW95] R. Corless, P. Gianni, B. Trager and S. Watt, *The Singular Value Decomposition for Polynomial Systems*, Proc. of ISSAC '95, ACM, 1995, 195–207.

[EGL97] I.Z. Emiris, A. Galligo and H. Lombarbi, *Certified Approximate Univariate GCDs*, J. Pure and Appl. Alge., **117&118** (1997), 229–251.

[GKMYZ04] S. Gao, E. Kaltofen, J.P. May, Z. Yang and L. Zhi, *Approximate Factorization of Multivariate Polynomials via Differential Equations*, Proc. of ISSAC '04, ACM, 2004, 167–174.

[GL89] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1989.

[Kal04] E. Kaltofen, `http://www4.ncsu.edu/~kaltofen/software/appfac/issac04_mws/`, 2004.

[KS97] F. Kako and T. Sasaki, *Proposal of "Effective Floating-point Number" for Approximate Algebraic Computation*, Preprint of Tsukuba Univ., 1997.

[LZN00] K. Li, L. Zhi and M-T. Noda, *Solving Approximate GCD of Multivariate Polynomial by Maple/Matlab/C Combination*, Proc. of ATCM 2000, World Scientific, 2000, 492–499.

[Matlab] MATLAB Application Program Interface Guide, The Math Works, Inc.

[MY73] J. Moses and D.Y.Y. Yun, *The EZGCD Algorithm*, Proc. of ACM National Conference, ACM, 1973, 159–166.

[ONS91] M. Ochi, M-T. Noda and T. Sasaki, *Approximate Greatest Common Divisor of Multivariate Polynomials and Its Application to Ill-Conditioned Systems of Algebraic Equations*, J. Inform. Proces., **14** (1991), 292–300.

[SN89] T. Sasaki and M-T. Noda, *Approximate Square-free Decomposition and Root-finding of Ill-conditioned Algebraic Equations*, J. Inform. Proces., **12** (1989), 159–168.

[SS92] T. Sasaki and M. Suzuki, *Three New Algrithms for Multivariate Polynomial GCD*, J. Symb. Comput., **13** (1992), 395–411.

[SY98] T. Sasaki and S. Yamaguchi, *An Analysis of Cancellation Error in Multivariate Hensel Construction with Floating-point Number Arithmetic*, Proc. of ISSAC '98, ACM, 1998, 1–8.

[Wan80] P.S. Wang, *The EEZ-GCD Algorithm*, SIGSAM Bulletin **14** (1980), 50–60.

[YZ05] T.Y. Li and Z. Zeng, *A Rank-Revealing Method with Updating, Downdating, and Applications*, SIAM J. Matrix Anal. Appl., **26** (2005), no. 4, 918–946.

[ZD04] Z. Zeng and B.H. Dayton, *The Approximate GCD of Inexact Polynomials Part I I: A Multivariate Algorithm*, Proc. of ISSAC '04, ACM, 2004, 320–327.

[Zen04] Z. Zeng, mvGCD – `http://www.neiu.edu/~zzeng/polynmat.htm`, 2004.

[Zhi04] L. Zhi, `http://www.mmrc.iss.ac.cn/~lzhi/Research/issac04_mws.html`, 2004.

[ZN00] L. Zhi and M-T. Noda, *Approximate GCD of Multivariate Polynomials*, Proc. of ASCM 2000, World Scientific, 2000, 9–18.

[ZY04] L. Zhi and Z. Yang, *Computing Approximate GCD of Multivariate Polynomials by Structure Total Least Norm*, MM Research Preprint, MMRC, AMSS, Academia, Sinica, 2004, No. 23, 388–401.

Masaru Sanuki
Graduate School of Pure and Applied Sciences
University of Tsukuba
Tsukuba-Shi, Ibaraki 305-8571, Japan
e-mail: `sanuki@math.tsukuba.ac.jp`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 69–83
© 2007 Birkhäuser Verlag Basel/Switzerland

Structured Low Rank Approximation of a Sylvester Matrix

Erich Kaltofen, Zhengfeng Yang and Lihong Zhi

Abstract. The task of determining the approximate greatest common divisor (GCD) of univariate polynomials with inexact coefficients can be formulated as computing for a given Sylvester matrix a new Sylvester matrix of lower rank whose entries are near the corresponding entries of that input matrix. We solve the approximate GCD problem by a new method based on structured total least norm (STLN) algorithms, in our case for matrices with Sylvester structure. We present iterative algorithms that compute an approximate GCD and that can certify an approximate ϵ-GCD when a tolerance ϵ is given on input. Each single iteration is carried out with a number of floating point operations that is of cubic order in the input degrees. We also demonstrate the practical performance of our algorithms on a diverse set of univariate pairs of polynomials.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 65K10.

Keywords. Sylvester matrix, approximate greatest common divisor, structured total least norm, hybrid symbolic/numeric algorithm.

1. Introduction

The problem of perturbation errors in the scalars of the inputs to a symbolic computation task has been studied extensively in the recent past, giving rise of the subject of hybrid symbolic/numeric algorithms. Approximate GCDs and factors have been at the center of investigations. One can formulate the algorithm specifications as an optimization problem without appealing to floating point arithmetic [11, 5]. In the GCD case one has the following.

This research was supported in part by the National Science Foundation of the USA under Grants CCR-0305314 and CCF-0514585 (Kaltofen) and OISE-0456285 (Kaltofen, Yang and Zhi), and by NKBRPC (2004CB318000) and the Chinese National Natural Science Foundation under Grant 10401035 (Yang and Zhi).

Problem 1.1. *Input are two univariate polynomials f, $g \in \mathbb{C}[x]$ with degree* $\deg(f) = m$ *and* $\deg(g) = n$. *For a positive integer k with $k \le \min(m, n)$, we wish to compute $\triangle f$, $\triangle g \in \mathbb{C}[x]$ such that* $\deg(\triangle f) \le m$, $\deg(\triangle g) \le n$, $\deg(\mathrm{GCD}(f+\triangle f, g+\triangle g)) \ge k$ *and such that $\|\triangle f\|_2^2 + \|\triangle g\|_2^2$ is minimized.*

When $k = 1$, a polynomial time solution is presented in [18]; see also [5, Sect. 2.6]. One may restrict the above problem to polynomials with entirely real coefficients. Several authors assume that an error estimate ϵ is also input and either output a pair $\triangle f$, $\triangle g$ with $\|\triangle f\|_2^2 + \|\triangle g\|_2^2 \le \epsilon$, yielding an ϵ-GCD equal $\mathrm{GCD}(f+\triangle f, g+\triangle g)$, or prove that no such pair exists or output “undecided,” the latter when the used numerical techniques cannot settle the problem.

The computation of approximate GCDs of univariate polynomials has been extensively studied [27, 21, 5, 7, 17, 10, 2, 22, 26, 32, 6, 30, 31]. The singular value decomposition (SVD) of the Sylvester matrix derived from the input polynomials is used in [5, 7, 32, 6, 8, 31] to deduce approximate GCDs. By dropping insignificant singular values, the SVD yields a nearby matrix of lower rank, but that matrix has no longer the Sylvester structure. The approximate GCD can be found by additional manipulation, for example from the singular vectors.

Here we propose to approximate the given Sylvester matrix with a rank deficient matrix that also has Sylvester structure, which is an instance of the class of *structure preserving* total least squares problems. There are several methods at our disposal, and we have tested the STLN (structured total least norm) algorithm [23] and the iterated projection algorithm in [4]. STLN is an efficient method for obtaining an approximate solution $(A+E)X = B+H$ to an overdetermined linear system $AX \approx B$, preserving the given linear structure in the minimal perturbation $[E\, H]$. We show how to solve Problem 1.1, at least for a local minimum, by applying STLN with L_2 norm to a submatrix of the Sylvester matrix. The algorithm in [4] projects the nearest rank deficient matrix by imposing Sylvester structure, thus destroying rank deficiency. Then it repeats the SVD/projection steps on the new Sylvester matrices.

We achieve excellent performance of STLN on our test cases, requiring only a handful of iterations and yielding a backward error that is comparable and better than earlier algorithms. For instance, our STLN-based approximations can have a relative backward error that is about 10 times smaller than the one achieved by the SVD-Gauss Newton approach [30]. In contrast, the algorithm in [4] does not perform well, exhibiting slow convergence similarly as experienced earlier on the factoring problem.

The organization of this paper is as follows. In Sect. 2, we introduce some notations and discuss the equivalence between the GCD problems and the low rank approximation of a Sylvester matrix. In Sect. 3, we consider solving an overdetermined system with Sylvester structure based on STLN. In Sect. 4, we describe our approximate GCD based on STLN and discuss the achieved practical performance on a number of benchmark pairs of univariate polynomials. Furthermore, we compare our results with preceding work. We conclude in Sect. 5 with remarks on the complexity and the rate of convergence of our algorithms.

2. Preliminaries

We first shall prove that the minimization Problem 1.1 in Sect. 1 always has a solution, in contrast to general and structured total least norm problems [19, Sect. 3.1]. Let f, $g \in \mathbb{C}[x] \setminus \{0\}$ with degree $\deg(f) = m$ and $\deg(g) = n$, namely

$$f = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0, \quad a_m \neq 0,$$
$$g = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0, \qquad b_n \neq 0.$$

Theorem 2.1. *Let k be a positive integer with $k \le \min(m, n)$. There exist $\hat{f}$, $\hat{g} \in \mathbb{C}[x]$ with $\deg(\hat{f}) \le m$, $\deg(\hat{g}) \le n$ and $\deg \mathrm{GCD}(\hat{f}, \hat{g}) \ge k$ such that for all $\tilde{f}, \tilde{g} \in \mathbb{C}[x]$ with $\deg(\tilde{f}) \le m$, $\deg(\tilde{g}) \le n$ and $\deg \mathrm{GCD}(\tilde{f}, \tilde{g}) \ge k$ we have*

$$\|\hat{f} - f\|_2^2 + \|\hat{g} - g\|_2^2 \le \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2.$$

Proof. Let $h \in \mathbb{C}[x]$ be monic with $\deg(h) = k$ and let $u, v \in \mathbb{C}[x]$ with $\deg(u) \le m - k$ and $\deg(v) \le n - k$. For the real and imaginary parts of the coefficients of h (excluding the leading coefficient, which is set to 1), and of u and v we consider the continuous objective function

$$F(h, u, v) = \|uh - f\|_2^2 + \|vh - g\|_2^2.$$

We prove that the function has a value on a closed and bounded set (with respect to the Euclidean metric) of its real argument vector that is smaller than elsewhere. Hence the function attains, by Weierstrass's theorem, a global minimum. Consider $\bar{f} = a_m x^m$ and $\bar{g} = b_n x^n$, which have a GCD of degree $\ge k$. Clearly, any triple h, u, v with $F(h, u, v) > \|\bar{f} - f\|_2^2 + \|\bar{g} - g\|_2^2$ can be discarded. So the coefficients of uh and vh can be bounded from above, and by any polynomial factor coefficient bound, so can the coefficients of h, u, v (provided $\|u\|_2$ or $\|v\|_2$ are bounded away from zero for monic h; see [16, Sect. 1.2] for more detail). Thus the domain of the function $F(h, u, v)$ can be restricted to a sufficiently large ball. It remains to exclude $u = v = 0$ as the minimal solution. We have $F(h, 0, 0) = \|f\|_2^2 + \|g\|_2^2 > \|\bar{f} - f\|_2^2 + \|\bar{g} - g\|_2^2$. □

Remark 2.2. The above theorem 2.1 remains valid when one restricts the input and perturbed polynomials to have real coefficients. We note that for real input polynomials the optimal complex solution may be nearer than the optimal real solution. In fact, for the pair $f = x^2 + 1$ and $g = x^2 + 2$, the optimal real solution for $k = 1$ is $\hat{f} = 0.723598\,x^2 + 1.170810$ and $\hat{g} = 1.170822\,x^2 + 1.894436$ with $\|\hat{f} - f\|_2^2 + \|\hat{g} - g\|_2^2 = 0.145898$, while there is a nearer pair of complex polynomials with a common root, namely $\hat{\hat{f}} = 0.81228x^2 - 0.14813\,ix + 1.1169$ and $\hat{\hat{g}} = 1.1220x^2 + 0.096263\,ix + 1.9240$ with $\|\hat{\hat{f}} - f\|_2^2 + \|\hat{\hat{g}} - g\|_2^2 = 0.1007615$. Moreover, in such a case, the nearest pair is never unique. The second solution is the complex conjugate of $\hat{\hat{f}}$ and $\hat{\hat{g}}$. A real example with an ambiguous real nearest pair with a GCD is $f = x^2 - 2$ and $g = x^2 - 1$. We note that ambiguity of approximate solutions was already noted in [33, 12]. □

Now suppose $S(f, g)$ is the Sylvester matrix of f and g. It is well-known that the degree of GCD of f and g is equal to the rank deficiency of S; we have

$$\min_{\deg(\mathrm{GCD}(\tilde{f},\tilde{g}))\geq k} \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2$$

$$\iff \min_{\mathrm{rank}(\tilde{S})\leq \tilde{n}+\tilde{m}-k} \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2 \tag{2.1}$$

where $\tilde{S}$ is the Sylvester matrix generated by $\tilde{f}$ and $\tilde{g}$, with $\deg \tilde{f} = \tilde{m} \leq m$ and $\deg \tilde{g} = \tilde{n} \leq n$. Note that for $\tilde{f} = 0$ or $\tilde{g} = 0$, the Sylvester matrix $\tilde{S}$ is not defined. If one polynomial is zero, we shall assume that $\mathrm{rank}(\tilde{S}) \leq \tilde{n} + \tilde{m} - k$ is satisfied since then GCD is the other non-zero polynomial. If both $\tilde{f} = \tilde{g} = 0$, we shall assume $\mathrm{rank}(\tilde{S}) = \infty$, as that solution is always sub-optimal (see proof of Theorem 2.1).

The k-th Sylvester matrix $S_k \in \mathbb{C}^{(m+n-k+1)\times(m+n-2k+2)}$ is a submatrix of S obtained by deleting the last $k-1$ rows of S and the last $k-1$ columns of coefficients of f and g separately in S.

$$S_k = \left[\begin{array}{ccc|ccc} a_m & & & b_n & & \\ a_{m-1} & \ddots & & b_{n-1} & \ddots & \\ \vdots & \ddots & a_m & \vdots & \ddots & b_n \\ a_0 & & a_{m-1} & b_0 & & b_{n-1} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & a_0 & & & b_0 \end{array}\right],$$
$$\underbrace{\qquad\qquad\qquad}_{n-k+1} \quad \underbrace{\qquad\qquad\qquad}_{m-k+1}$$

For $k = 1$, $S_1 = S$ is the Sylvester matrix. In paper [8], we know the strong relationship between the Sylvester matrix S and its k-th submatrix S_k.

Theorem 2.3 ([8]). *Given univariate polynomials f, $g \in \mathbb{C}[x]$, $\deg(f)=m$, $\deg(g)= n$ and $1 \leq k \leq \min(m,n)$. $S(f,g)$ is the Sylvester matrix of f and g, S_k is the k-th Sylvester matrix of f and g. Then the following statements are equivalent:*

(a) $\mathrm{rank}(S) \leq m + n - k$;
(b) *Rank deficiency of S_k is greater than or equal to one.*

Having the above theorem, the formulation (2.1) can be transformed into:

$$\min_{\deg(\mathrm{GCD}(\tilde{f},\tilde{g}))\geq k} \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2$$

$$\iff \min_{\dim\ \mathrm{Nullspace}(\tilde{S}_k)\geq 1} \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2, \tag{2.2}$$

where $\tilde{S}_k$ is the k-th Sylvester matrix generated by $\tilde{f}$ and $\tilde{g}$, with $\deg \tilde{f} \le m$ and $\deg \tilde{g} \le n$.

If we solve the following overdetermined system using STLN [25, 24, 23, 19]

$$A_k\, \mathbf{x} \approx \mathbf{b}_k, \tag{2.3}$$

for $S_k = [\mathbf{b}_k\, A_k]$, where $\mathbf{b}_k$ is the first column of S_k and A_k are the remainder columns of S_k, then we obtain a minimal perturbation $[\mathbf{h}_k\, E_k]$ of Sylvester structure such that

$$\mathbf{b}_k + \mathbf{h}_k \in \text{Range}(A_k + E_k).$$

Therefore, $\tilde{S}_k = [\mathbf{b}_k + \mathbf{h}_k,\ A_k + E_k]$ is a solution with Sylvester structure (provided the highest order coefficients remain non-zero) and dim Nullspace$(\tilde{S}_k) \ge 1$.

The reason why we choose the first column to form the overdetermined system (2.3) can be seen from the following example and theorem.

Example 1. Suppose we are given two polynomials

$$f = x^2 + x = x\,(x+1),$$
$$g = x^2 + 4x + 3 = (x+3)\,(x+1).$$

S is the Sylvester matrix of f and g:

$$S = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 4 & 1 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

The rank deficiency of S is 1. We partition S in two ways: $S = [\hat{A}_1\, \hat{\mathbf{b}}_1] = [\bar{\mathbf{b}}_1\, \bar{A}_1]$, where $\hat{\mathbf{b}}_1$ is the last column of S, whereas $\bar{\mathbf{b}}_1$ is the first column of S.

The overdetermined system

$$\hat{A}_1 \mathbf{x} = \hat{\mathbf{b}}_1$$

has no solution, while the system

$$\bar{A}_1 \mathbf{x} = \bar{\mathbf{b}}_1$$

has an exact solution as $\mathbf{x} = [-3, 1, 0]^T$.

Theorem 2.4. *Given univariate polynomials $f, g \in \mathbb{C}[x]$ with $\deg(f) = m$, $\deg(g) = n$ and a positive integer $k \le \min(m, n)$. Suppose S_k is the k-th Sylvester matrix of f and g. Partition $S_k = [\mathbf{b}_k\, A_k]$, where $\mathbf{b}_k$ is the first column of S_k and A_k consists of the last $n + m - 2k + 1$ columns of S_k. Then we have*

$$\text{dim Nullspace}(S_k) \ge 1 \iff A_k \mathbf{x} = \mathbf{b}_k \textit{ has a solution.} \tag{2.4}$$

Proof. "$\Longleftarrow$": Let $A_k \mathbf{x} = \mathbf{b}_k$ have a solution, then $\mathbf{b}_k \in \text{Range}(A_k)$. Since $\mathbf{b}_k$ is the first column of S_k, the rank deficiency of $S_k = [\mathbf{b}_k\, A_k]$ is at least 1.

"$\Longrightarrow$:" Suppose the rank deficiency of $S_k = [\mathbf{b}_k\, A_k]$ is at least 1. Multiplying the vector $[x^{n+m-k}, \ldots, x, 1]$ to the two sides of the equation $A_k \mathbf{x} = \mathbf{b}_k$, it turns out to be

$$[x^{n-k-1} f, \ldots, f, x^{m-k} g, x^{m-k-1} g, \ldots, g]\mathbf{x} = x^{n-k} f. \tag{2.5}$$

The solution $\mathbf{x}$ of (2.5) corresponds to the coefficients of polynomials u, v, with $\deg(u) \leq n-k-1, \deg(v) \leq m-k$ and satisfy

$$x^{n-k} f = u\, f + v\, g.$$

Let $d = \mathrm{GCD}(f, g)$, $f_1 = f/d, g_1 = g/d$. Since dim Nullspace$(S_k) \geq 1$, we have $\deg(d) \geq k$ and $\deg(f_1) \leq m-k, \deg(g_1) \leq n-k$. Dividing x^{n-k} by g_1, we have a quotient q and a remainder p such that

$$x^{n-k} = q\, g_1 + p,$$

where $\deg(q) \leq \deg(d) - k, \deg(p) \leq n-k-1$. Now we can check that

$$u = p,\ v = q\, f_1,$$

are solutions of (2.5), since $\deg(u) \leq n-k-1$,

$$\deg(v) = \deg(q) + \deg(f_1) \leq \deg(d) - k + \deg(f_1) \leq m-k,$$

and

$$v\, g + u\, f = f_1\, q\, d\, g_1 + p\, f = f\, q\, g_1 + f\, p = f\, x^{n-k}. \qquad \square$$

Next, we show that for any given Sylvester matrix, when all the elements are allowed to be perturbed, it is always possible to find matrices $[\mathbf{h}_k\, E_k]$ with k-Sylvester structure (implying that the leading entries are non-zero) such that $\mathbf{b}_k + \mathbf{h}_k \in \mathrm{Range}(A_k + E_k)$, where $\mathbf{b}_k$ is the first column of S_k and A_k are the remainder columns of S_k.

Theorem 2.5. *Given the integers m, n and k, $k \leq \min(m, n)$, then there exists a Sylvester matrix $S \in \mathbb{C}^{(m+n)\times(m+n)}$ with rank $m+n-k$.*

Proof. For all m and n, we always can construct polynomials f, $g \in \mathbb{C}[x]$ such that $\deg(f) = m$, $\deg(g) = n$, and the degree of $\mathrm{GCD}(f, g)$ is k. Hence S is the Sylvester matrix generating by f, g and its rank is $m+n-k$. $\square$

Corollary 2.6. *Given the positive integers m, n, $k \leq \min(m, n)$, and k-th Sylvester matrix $S_k = [\mathbf{b}_k\, A_k]$, where $A_k \in \mathbb{C}^{(m+n-k+1)\times(m+n-2k+1)}$ and $\mathbf{b}_k \in \mathbb{C}^{(m+n-k+1)\times 1}$, it is always possible to find a perturbation $[\mathbf{h}_k\, E_k]$ of k-th Sylvester structure such that $\mathbf{b}_k + \mathbf{h}_k \in \mathrm{Range}(A_k + E_k)$.*

3. STLN for Overdetermined Systems with Sylvester Structure

In this section, we illustrate how to solve the overdetermined system

$$A_k\, \mathbf{x} \approx \mathbf{b}_k, \tag{3.1}$$

where $A_k \in \mathbb{C}^{(m+n-k+1)\times(m+n-2k+1)}$ and $\mathbf{b}_k \in \mathbb{C}^{(m+n-k+1)\times 1}$, $S_k = [\mathbf{b}_k\, A_k]$ is the k-th Sylvester matrix. According to Theorem 2.5 and Corollary 2.6, there always exists k-th Sylvester structure perturbation $[\mathbf{h}_k\, E_k]$ such that $(\mathbf{b}_k + \mathbf{h}_k) \in$ *Range*$(A_k + E_k)$. In the following, we illustrate how to find the minimum solution using STLN.

First, the Sylvester-structure preserving perturbation $[\mathbf{h}_k\, E_k]$ of S_k

$$[\mathbf{h}_k\, E_k] = \left[\begin{array}{cccccc} z_1 & & & z_{m+2} & & \\ z_2 & \ddots & & z_{m+3} & \ddots & \\ \vdots & \ddots & z_1 & \vdots & \ddots & z_{m+2} \\ z_{m+1} & & z_2 & z_{m+n+2} & & z_{m+3} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & z_{m+1} & & & z_{m+n+2} \end{array}\right]$$
$$\qquad\qquad \underbrace{\qquad\qquad\qquad}_{n-k+1} \quad \underbrace{\qquad\qquad\qquad}_{m-k+1}$$

can be represented by a vector $\mathbf{z} \in \mathbb{C}^{(m+n+2)\times 1}$:

$$\mathbf{z} = [z_1, z_2, \ldots, z_{m+n+1}, z_{m+n+2}]^T.$$

Since $\mathbf{h}_k$ is the first column of the above matrix, we can define a matrix P_k as

$$P_k = \left[\begin{array}{cc} \mathbf{I}_{m+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array}\right] \in \mathbb{C}^{(m+n-k+1)\times(m+n+2)}, \tag{3.2}$$

where $\mathbf{I}_{m+1}$ is a $(m+1)\times(m+1)$ identity matrix, such that $\mathbf{h}_k = P_k\mathbf{z}$.

We solve the equality-constrained least squares problem:

$$\min_{\mathbf{z},\,\mathbf{x}} \|\mathbf{z}\|_2, \text{ subject to } \mathbf{r} = 0, \tag{3.3}$$

where the structured residual $\mathbf{r}$ is

$$\mathbf{r} = \mathbf{r}(\mathbf{z}, \mathbf{x}) = \mathbf{b}_k + \mathbf{h}_k - (A_k + E_k)\mathbf{x}.$$

We do not know if the above STLN problem always has a solution. But even if that were the case (cf. [19, Theorem 3.1.2]), the optimal solution may not correspond to a nearest GCD pair, as it may correspond to polynomials of smaller degrees that, for instance, remain relatively prime. The structured minimization problem (3.3) can be solved by using the penalty method in [1], transforming (3.3) into:

$$\min_{\mathbf{z}\,,\mathbf{x}} \left\| \begin{array}{c} w\mathbf{r}(\mathbf{z}, \mathbf{x}) \\ \mathbf{z} \end{array} \right\|_2, \qquad w \gg 1, \tag{3.4}$$

where w is a large penalty value between 10^8 and 10^{10}. It is shown in [1, 28] that an algorithm based on Givens rotations produces accurate results regardless of row sorting and even with extremely large penalty values.

Following [24, 23], we use a linear approximation to $\mathbf{r}(\mathbf{z}, \mathbf{x})$ to solve the minimization problem. Let $\triangle\mathbf{z}$ and $\triangle\mathbf{x}$ represent a small change in $\mathbf{z}$ and $\mathbf{x}$ respectively, and $\triangle E_k$ represents the corresponding change in E_k. Then the first order approximation to $\mathbf{r}(\mathbf{z}+\triangle\mathbf{z}\,,\mathbf{x}+\triangle\mathbf{x})$ is

$$\begin{aligned}\mathbf{r}(\mathbf{z}+\triangle\mathbf{z}\,,\mathbf{x}+\triangle\mathbf{x}) &= \mathbf{b}_k + P_k(\mathbf{z}+\triangle\mathbf{z}) - (A_k+E_k+\triangle E_k)(\mathbf{x}+\triangle\mathbf{x})\\ &\approx \mathbf{b}_k + P_k\mathbf{z} - (A_k+E_k)\mathbf{x} + P_k\triangle\mathbf{z} - (A_k+E_k)\triangle\mathbf{x} - \triangle E_k\mathbf{x}\\ &= \mathbf{r} + P_k\triangle\mathbf{z} - (A_k+E_k)\triangle\mathbf{x} - \triangle E_k\mathbf{x}.\end{aligned}$$

We introduce a matrix of Sylvester structure $Y_k \in \mathbb{C}^{\mu\times\nu}$, where $\mu = m+n-k+1$ and $\nu = m+n+2$, such that

$$Y_k\,\triangle\mathbf{z} = \triangle E_k\,\mathbf{x} \quad \text{with} \quad \mathbf{x} = [x_1\,, x_2,\, \ldots, x_{m+n-2k+1}]^T. \tag{3.5}$$

Now (3.4) can be approximated by

$$\min_{\triangle\mathbf{x},\,\triangle\mathbf{z}} \left\| \begin{bmatrix} w(Y_k - P_k) & w(A_k+E_k) \\ \mathbf{I}_{m+n+2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \triangle\mathbf{z} \\ \triangle\mathbf{x} \end{bmatrix} + \begin{bmatrix} -w\mathbf{r} \\ \mathbf{z} \end{bmatrix} \right\|_2. \tag{3.6}$$

In the following, we propose a new method to construct the matrix Y_k. Suppose f, g, E, $\mathbf{z}$ and $\mathbf{x}$ are given above. Multiplying the vector

$$\mathbf{v} = [x^{m+n-k},\, x^{m+n-k-1}, \ldots,\, x^2,\, x^1,\, 1] \in \mathbb{C}[x]^{m+n-k+1}$$

to the two sides of (3.5), we obtain the polynomial identity

$$\mathbf{v}\,Y_k\,\mathbf{z} = \mathbf{v}\,E_k\,\mathbf{x}.$$

For $\begin{bmatrix}0\\ \mathbf{x}\end{bmatrix}$ we obtain

$$\mathbf{v}\,E_k\,\mathbf{x} = \mathbf{v}\,[\mathbf{h}_k, E_k]\,\hat{\mathbf{x}} = \hat{g}_1\hat{u}_1 + \hat{g}_2\hat{u}_2, \tag{3.7}$$

where $\hat{g}_1$ is the polynomial of degree m, generated by the subvector of $\mathbf{z}$:

$$[z_1,\ z_2,\ \ldots,\ z_{m+1}],$$

$\hat{g}_2$ is the polynomial of degree n, generated by the subvector of $\mathbf{z}$:

$$[z_{m+2},\ z_{m+3},\ \ldots,\ z_{m+n+2}],$$

$\hat{u}_1$ is the polynomial of degree $n-k-1$, generated by the subvector of $\hat{\mathbf{x}}$:

$$[0,\ x_1,\ x_2,\ \ldots,\ x_{n-k}],$$

$\hat{u}_2$ is the polynomial of degree $m-k$, generated by the subvector of $\hat{\mathbf{x}}$:

$$[x_{n-k+1},\ x_{n-k+2},\ \ldots,\ x_{m+n-2k+1}].$$

Y_k is the coefficient matrix formed from the above linear system (3.7) with respect to powers of x and the variables z_i:

$$Y_k = \begin{bmatrix} 0 & & & x_{n+1-k} & & \\ x_1 & \ddots & & x_{n+2-k} & \ddots & \\ \vdots & \ddots & 0 & \vdots & \ddots & x_{n+1-k} \\ x_{n-k} & & x_1 & x_{m+n+1-2k} & & x_{n+2-k} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & x_{n-k} & & & x_{m+n+1-2k} \end{bmatrix}.$$

$$\underbrace{\qquad\qquad\qquad}_{m+1} \qquad \underbrace{\qquad\qquad\qquad\qquad}_{n+1}$$

Example 2. Suppose $m = n = 3$, $k = 2$; then $S_2 = [\mathbf{b}_2 \; A_2]$, where

$$A_2 = \begin{bmatrix} 0 & b_3 & 0 \\ a_3 & b_2 & b_3 \\ a_2 & b_1 & b_2 \\ a_1 & b_0 & b_1 \\ a_0 & 0 & b_0 \end{bmatrix}, \; \mathbf{b}_2 = \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \\ 0 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$Y_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 \\ x_1 & 0 & 0 & 0 & x_3 & x_2 & 0 & 0 \\ 0 & x_1 & 0 & 0 & 0 & x_3 & x_2 & 0 \\ 0 & 0 & x_1 & 0 & 0 & 0 & x_3 & x_2 \\ 0 & 0 & 0 & x_1 & 0 & 0 & 0 & x_3 \end{bmatrix}.$$

It is easy to see that the coefficient matrix in (3.6) is also of block Toeplitz structure. We could apply fast least squares method to solve it quickly. Preliminary results on that are reported in [20].

4. Approximate GCD Algorithm and Experiments

The following algorithm is designed for finding an approximate solution to Problem 1.1.

Algorithm AppSylv-k

Input — A Sylvester matrix S generated by two polynomials f, $g \in \mathbb{C}[x]$ of total degrees $m \geq n$ respectively, an integer $1 \leq k \leq n$ and a tolerance *tol*.

Output — Polynomials $\tilde{f}$ and $\tilde{g}$ with dim Nullspace$(S(\tilde{f},\tilde{g})) \geq k$ and the Euclidean distance $\|\tilde{f}-f\|_2^2 + \|\tilde{g}-g\|_2^2$ is reduced to a minimum.

```
1. Form the k-th Sylvester matrix $S_k$, choose the first column of $S_k$ as $\mathbf{b}_k$, and
   $A_k$ be the remainder columns of $S_k$. Let $E_k = \mathbf{0}$, $\mathbf{h}_k = \mathbf{0}$.
2. Compute $\mathbf{x}$ from $\min \|A_k\mathbf{x} - \mathbf{b}_k\|_2$ and $\mathbf{r} = \mathbf{b}_k - A_k\mathbf{x}$. Form $P_k$ and $Y_k$ as
   shown in the above sections.
3. Repeat
   a) $\min_{\triangle\mathbf{x},\,\triangle\mathbf{z}} \left\| \begin{bmatrix} w(Y_k - P_k) & w(A_k + E_k) \\ \mathbf{I}_{m+n+2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \triangle\mathbf{z} \\ \triangle\mathbf{x} \end{bmatrix} + \begin{bmatrix} -w\mathbf{r} \\ \mathbf{z} \end{bmatrix} \right\|_2.$
   b) Set $\mathbf{x} = \mathbf{x} + \triangle\mathbf{x}$, $\mathbf{z} = \mathbf{z} + \triangle\mathbf{z}$.
   c) Construct the matrix $E_k$ and $\mathbf{h}_k$ from $\mathbf{z}$, and $Y_k$ from $\mathbf{x}$. Set $A_k =$
      $A_k + E_k$, $\mathbf{b}_k = \mathbf{b}_k + \mathbf{h}_k$, $\mathbf{r} = \mathbf{b}_k - A_k\mathbf{x}$.
      until $(\|\triangle\mathbf{x}\|_2 \leq tol$ and $\|\triangle\mathbf{z}\|_2 \leq tol)$.
4. Output the polynomials $\tilde{f}$ and $\tilde{g}$ formed from $\mathbf{b}_k$ and $A_k$.
```

Given a tolerance ϵ, the algorithm AppSylv-k can be used to compute an ϵ-GCD of polynomials f and g with degrees $m \geq n$ respectively. The method starts with $k = n \leq m$, using AppSylv-k to compute the minimum $\mathcal{N} = ||\tilde{f} - f||_2 + ||\tilde{g} - g||_2$ with rank$(S(\tilde{f},\tilde{g})) \leq m + n - k$. If $\mathcal{N} < \epsilon$, then we can compute the ϵ-GCD from the matrix $S_k(\tilde{f},\tilde{g})$ [8, 30]; Otherwise, we reduce k by one and repeat the AppSylv-k algorithm. Another method tests the degree of ϵ-GCD by computing the singular value decomposition of Sylvester matrix $S(f,g)$, find the upper bound degree r of the ϵ-GCD as shown in [5, 7]. So we can start with $k = r$ rather than $k = n$ to compute the certified ϵ-GCD of the highest degree.

Example 3. The following example is given in Karmarkar and Lakshman's paper [18]. We wish to find the minimal polynomial perturbations $\triangle f$ and $\triangle g$ of

$$f = x^2 - 6x + 5 = (x-1)(x-5),$$
$$g = x^2 - 6.3x + 5.72 = (x-1.1)(x-5.2),$$

such that the polynomials $f + \triangle f$ and $g + \triangle g$ have a common root. We consider this problem in two cases: the leading coefficients can be perturbed and the leading coefficients can not be perturbed.

Case 1: The leading coefficients can be perturbed. Applying the algorithm App Sylv-k to f, g with $k = 1$ and $tol = 10^{-3}$, after three iterations, we obtain the polynomials $\tilde{f}$ and $\tilde{g}$:

$$\tilde{f} = 0.9850x^2 - 6.0029x + 4.9994,$$
$$\tilde{g} = 1.0150x^2 - 6.2971x + 5.7206,$$

with a distance

$$\mathcal{N} = \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2 = 0.0004663.$$

The common root of the $\tilde{f}$ and $\tilde{g}$ is 5.09890429.

Case 2: The leading coefficients can not be perturbed, i.e., the first and fourth terms of $\mathbf{q}$ are fixed as one. Running the algorithm AppSylv-k for $k = 1$ and $tol = 10^{-3}$, after three iterations, we get the polynomials $\tilde{f}$ and $\tilde{g}$:

$$\tilde{f} = x^2 - 6.0750x + 4.9853,$$
$$\tilde{g} = x^2 - 6.2222x + 5.7353,$$

with minimum distance as

$$\mathcal{N} = \|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2 = 0.01213604583.$$

The common root of $\tilde{f}$ and $\tilde{g}$ is 5.0969478.

In the paper [18] the perturbed polynomials are restricted to be monic. The minimum perturbation Karmarkar and Lakshman obtained is 0.01213605293, which corresponds to the perturbed common root 5.096939087.

Remark 4.1. The above algorithm will for real inputs compute the optimal pair over the reals only. As stated in Remark 2.2 even for real inputs the optimal complex pair may have complex coefficients. The following change in the initialization in step 2 can accomplish that:

$2^{\mathbb{C}}$. Compute $\mathbf{x}$ from $\min \|(A_k+\delta A_k)\mathbf{x}-(\mathbf{b}_k+\delta \mathbf{b}_k\|_2$ and $\mathbf{r} = \mathbf{b}_k+\delta\mathbf{b}_k-A_k-\delta A_k\mathbf{x}$, where δA_k and $\delta\mathbf{b}_k$ are structured perturbations of random purely imaginary complex noise. Form P_k and Y_k as shown in the above sections.

For the first iteration in step 3 we use the original real input coefficients. The optimal complex solutions in Remark 2.2 could be found by adding random noise of magnitute 10^{-2}.

We have also tested our algorithm on inputs where both polynomials have small leading coefficients and observed that the method can produce valid results provided the “tails,” of the polynomials, i.e., the parts without the leading coefficients, are approximately relative prime. We present an example.

$$f = .1000000000 \cdot 10^{-9}x^2 + x,$$
$$g = .1000000000 \cdot 10^{-9}x^2 + x + 1.$$

The roots of f are -10^{10}, 0, the roots of g are -9999999999, -1. We compute two polynomials $\bar{f}$ and $\bar{g}$ by our algorithm:

$$\bar{f} = .100000000005000000 \cdot 10^{-9}x^2 + 1.x,$$
$$\bar{g} = .99999999994999994 \cdot 10^{-10}x^2 + 1.x + 1.$$

The roots of $\bar{f}$ are now -9999999999.5, 0 and the roots of $\bar{g}$ are now -9999999999.5, -1. The perturbation by our algorithm is

$$\|\bar{f} - f\|_2^2 + \|\bar{g} - g\|_2^2 = .500000000000000 \cdot 10^{-40}.$$

However, if the tails have a nearby GCD, the algorithm as stated does not find a good result due to the choice of $\mathbf{b}_k$ as the first column of the Sylvester matrix. That problem appears in a more general manner when applying our approach to multivariate approximate GCDs when some terms of maximal total degree can vanish in the nearest pair. The remedy is to determine the proper right side vector from the components of the first singular vector; see [14, 16] for more detail. □

In Table 1, we show the performance of our algorithm for computing ϵ-GCDs of univariate polynomials randomly generated in Maple 9 with *Digits* = 10. For every example, we use 50 random cases for each (m, n), and report the average over all results. For each example, the prime parts and GCD of two polynomials are constructed by choosing polynomials with random integer coefficients in the range $-10 \le c \le 10$, and then adding a perturbation. For noise we choose a relative tolerance 10^{-e}, then randomly choose a polynomial that has the same degree as the product, and coefficients in $[-10^e, 10^e]$. Finally, we scale the perturbation so that the relative error is 10^{-e}. In our test cases we set $e = 2$. Here m and n denote the total degrees of polynomials f and g; k is the degree of approximate GCD of f and g; "it. (Chu)" is the number of the iterations needed by Chu's method [4]; whereas "it. (STLN)" denotes the number of iterations by AppSylv-k algorithm; "error (Zeng)" denotes the perturbation $\|\bar{f} - f\|_2^2 + \|\bar{g} - g\|_2^2$ computed by Zeng's algorithm [30]; whereas "error (STLN)" is the minimal perturbation $\|\tilde{f} - f\|_2^2 + \|\tilde{g} - g\|_2^2$ computed by AppSylv-k algorithm; σ_k and $\tilde{\sigma}_k$ are the last k-th singular values of $S(f, g)$ and $S(\tilde{f}, \tilde{g})$, respectively. Riemannian SVD has been considered in [3] for computing approximate GCDs. We would like to compare with their implementation in the future.

Table 1. Algorithm performance on benchmarks (univariate case)

Ex.	m, n	k	it. (Chu)	it. (STLN)	error (Zeng)	error (STLN)	σ_k	$\tilde{\sigma}_k$
1	$2, 2$	1	4.73	1.90	1.89e–4	2.87e–5	3.53e–3	10^{-9}
2	$3, 3$	2	8.49	1.98	1.36e–3	1.05e–4	8.21e–3	10^{-9}
3	$5, 4$	3	11.44	2.00	1.00e–3	1.25e–4	1.01e–2	10^{-9}
4	$5, 5$	3	13.64	2.00	7.43e–4	1.25e–4	9.57e–3	10^{-9}
5	$6, 6$	4	23.07	2.00	1.46e–3	1.41e–4	9.64e–3	10^{-9}
6	$8, 7$	4	32.64	2.00	6.53e–4	1.31e–4	8.04e–3	10^{-9}
7	$10, 10$	5	43.12	2.00	1.61e–3	2.01e–4	1.21e–2	10^{-9}
8	$14, 13$	7	58.16	2.00	1.23e–3	2.52e–4	1.51e–2	10^{-9}
9	$28, 28$	10	161.74	2.00	2.61e–3	3.41e–4	1.48e–2	10^{-10}
10	$65, 65$	15	633.64	2.00	6.19e–3	5.50e–4	1.90e–2	10^{-9}

5. Concluding Remarks

In this paper we present a practical and reliable way based on STLN to compute the approximate GCD of univariate polynomials. Note that the overall computational complexity of the algorithm AppSylv-k depends on the number of iterations needed for completing step 3 and the computational complexity of each iteration. If the starting values are good, then the iteration will converge quickly. This can be seen from the above table. For each iteration, Givens rotations are applied. Checking the size of the matrix involved in solving the minimization problem in step 3, we obtain that each iteration needs less than $(4\,m + 4\,n - k + 6)\,(2\,m + 2\,n - 2\,k + 3)^2$ operations. Since the matrices involved in the minimization problems are all structured matrix, they have low displacement rank. It would be possible to apply the fast algorithm to solve these minimization problems [20]. This would reduce the complexity of our algorithm to be only quadratic with respect to the degrees of the given polynomials.

Our methods can be generalized to several polynomials and to several variables [15, 16]. Moreover, as observed in [9], arbitrary linear equational constraints can be imposed on the coefficients of the input and perturbed polynomials. Such constraints can be used to preserve monicity and sparsity, but also relations among the input polynomials' coefficients [15, 16].

Acknowledgement

We thank the referees of this paper and its earlier version for their helpful remarks. A preliminary version of this paper appears in [13].

References

[1] A.A. Anda and H. Park, *Fast plane with dynamic scaling*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 162–174.

[2] B. Beckermann and G. Labahn, *When are two polynomials relatively prime?* J. Symbolic Comput., 26 (1998), pp. 677–689.

[3] B. Botting, M. Giesbrecht and J. May, *Using Riemannian SVD for problems in approximate algebra*, in Wang and Zhi [29], pp. 209–219. Distributed at the Workshop in Xi'an, China.

[4] M.T. Chu, R.E. Funderlic and R.J. Plemmons, *Structured low rank approximation*, Linear Algebra and Applications, 366 (2003), pp. 157–172.

[5] R.M. Corless, P.M. Gianni, B.M. Trager and S.M. Watt, *The singular value decomposition for polynomial systems*, in Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. (ISSAC '95), A. H. M. Levelt, ed., New York, N. Y., 1995, ACM Press, pp. 96–103.

[6] R.M. Corless, S.M. Watt and L. Zhi, *QR factoring to compute the GCD of univariate approximate polynomials*, IEEE Transactions on Signal Processing, 52 (2004), pp. 3394–3402.

[7] I.Z. Emiris, A. Galligo and H. Lombardi, *Certified approximate univariate GCDs*, J. Pure Applied Algebra, 117 & 118 (1996), pp. 229–251. Special Issue on Algorithms for Algebra.

[8] S. Gao, E. Kaltofen, J.P. May, Z. Yang and L. Zhi, *Approximate factorization of multivariate polynomials via differential equations*, in Proc. 2004 Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2004), J. Gutierrez, ed., New York, N. Y., 2004, ACM Press, pp. 167–174.

[9] M.A. Hitz and E. Kaltofen, *Efficient algorithms for computing the nearest polynomial with constrained roots*, in Proc. 1998 Internat. Symp. Symbolic Algebraic Comput. (ISSAC '98), O. Gloor, ed., New York, N. Y., 1998, ACM Press, pp. 236–243.

[10] V. Hribernig and H.J. Stetter, *Detection and validation of clusters of polynomials zeros*, J. Symbolic Comput., 24 (1997), pp. 667–681.

[11] E. Kaltofen, *Polynomial factorization 1987-1991*, in Proc. LATIN '92, I. Simon, ed., vol. 583 of Lect. Notes Comput. Sci., Heidelberg, Germany, 1992, Springer Verlag, pp. 294–313.

[12] E. Kaltofen and J. May, *On approximate irreducibility of polynomials in several variables*, in Proc. 2003 Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2003), J. R. Sendra, ed., New York, N. Y., 2003, ACM Press, pp. 161–168.

[13] E. Kaltofen, Z. Yang and L. Zhi, *Structured low rank approximation of a Sylvester matrix*, in Wang and Zhi [29], pp. 188–201. Distributed at the Workshop in Xi'an, China.

[14] E. Kaltofen, Z. Yang and L. Zhi, *Structured low rank approximation of a generalized Sylvester matrix*, in Proc. of the Seventh Asian Symposium on Computer Mathematics, S. Pae and H. Park, eds., Seoul, South Korea, 2005, Korea Institute for Advanced Study, pp. 219–222. Extended abstract.

[15] E. Kaltofen, Z. Yang and L. Zhi, *Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials*, in Proc. 2006 Internat. Symp. Symbolic Algebraic Comput. (ISSAC '06), J.-G. Dumas, ed., New York, N. Y., 2006, ACM Press, pp. 169–176.

[16] E. Kaltofen, Z. Yang and L. Zhi, *Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials.* Manuscript, 20 pages, Nov. 2006.

[17] N. Karmarkar and Y.N. Lakshman, *Approximate polynomial greatest common divisors and nearest singular polynomials*, in Proc. 1996 Internat. Symp. Symbolic Algebraic Comput. (ISSAC '96), Lakshman Y. N., ed., New York, N. Y., 1996, ACM Press, pp. 35–42.

[18] N.K. Karmarkar and Y.N. Lakshman, *On approximate GCDs of univariate polynomials*, J. Symbolic Comput., 26 (1998), pp. 653–666. Special issue of the JSC on Symbolic Numeric Algebra for Polynomials, S.M. Watt and H.J. Stetter, editors.

[19] P. Lemmerling, *Structured total least squares: analysis, algorithms and applications*, dissertation, Katholieke Universiteit Leuven, Belgium, 1999.

[20] B. Li, Z. Yang, and L. Zhi, *Fast low rank approximation of a Sylvester matrix by structured total least norm*, J. JSSAC (Japan Society for Symbolic and Algebraic Computation), 11 (2005), pp. 165–174.

[21] M.T. Noda and T. Sasaki, *Approximate GCD and its application to ill-conditioned algebraic equations*, J. Comput. Appl. Math., 38 (1991), pp. 335–351.

[22] V.Y. Pan, *Numerical computation of a polynomial GCD and extensions*, Information and computation, 167 (2001), pp. 71–85.

[23] H. Park, L. Zhang and J.B. Rosen, *Low rank approximation of a Hankel matrix by structured total least norm*, BIT, 39 (1999), pp. 757–779.

[24] J.B. Rosen, H. Park and J. Glick, *Total least norm formulation and solution for structured problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 110–128.

[25] J.B. Rosen, H. Park and J. Glick, *Structured total least norm for nonlinear problems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 14–30.

[26] M. Sasaki and T. Sasaki, *Polynomial remaider sequence and approximate GCD*, ACM SIGSAM Bulletin, 31 (2001), pp. 4–10.

[27] A. Schönhage, *Quasi-gcd computations*, Journal of Complexity, 1 (1985), pp. 118–137.

[28] G.W. Stewart, *On the asymptotic behavior of scaled singular value and QR decompositions*, Mathematics of Computation, 43 (1983), pp. 488–489.

[29] D. Wang and L. Zhi, eds., *Proc. 2005 International Workshop on Symbolic-Numeric Computation*, July 2005. Distributed at the Workshop in Xi'an, China.

[30] Z. Zeng, *The approximate GCD of inexact polynomials. Part I: A univariate algorithm*. Manuscript, 2004.

[31] Z. Zeng, *Computing multiple roots of inexact polynomials*, Math. Comput., 74 (2005), pp. 869–903.

[32] L. Zhi, *Displacement structure in computing approximate GCD of univariate polynomials*, in Proc. Sixth Asian Symposium on Computer Mathematics (ASCM 2003), Z. Li and W. Sit, eds., vol. 10 of Lecture Notes Series on Computing, Singapore, 2003, World Scientific, pp. 288–298.

[33] L. Zhi and W. Wu, *Nearest singular polynomial*, J. Symbolic Comput., 26 (1998), pp. 667–675. Special issue on Symbolic Numeric Algebra for Polynomials, S.M. Watt and H.J. Stetter, editors.

Erich Kaltofen
Department of Mathematics
North Carolina State University
Raleigh, North Carolina 27695-8205, USA
e-mail: `kaltofen@math.ncsu.edu`

Zhengfeng Yang and Lihong Zhi
Key Laboratory of Mathematics Mechanization
AMSS, Chinese Academy of Sciences
Beijing 100080, China
e-mail: `{zyang,lzhi}@mmrc.iss.ac.cn`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 85–97
© 2007 Birkhäuser Verlag Basel/Switzerland

Implementation of Fast Low Rank Approximation of a Sylvester Matrix

Bingyu Li, Zhuojun Liu and Lihong Zhi

Abstract. We describe and implement a fast algorithm for constructing structured low rank approximation of a Sylvester matrix. The fast algorithm is obtained by exploiting low displacement ranks of the involved structured matrices. We present detailed error analysis and experiments to show that the fast algorithm is stable.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 65F05.

Keywords. Sylvester matrix, displacement rank, generalized Schur algorithm, structured total least norm.

1. Introduction

The authors in [11] described a fast algorithm based on structured total least norm (STLN) [16, 14] for constructing structured low rank approximation of a Sylvester matrix and obtaining the nearest perturbed polynomials with exact GCD of degree not less than a given positive integer. This algorithm is of complexity $O((2m+2n-k+3)^2)$, where m, n, k are degrees of input polynomials and a given positive integer. The increased efficiency is obtained by exploiting low displacement ranks of the involved structured matrices in [10, 11]. However, since coefficient matrices appeared in the STLN method have large condition numbers, it is necessary to reduce error by choosing a suitable generator matrix for the fast algorithm. In this paper, we present a new generator pair of the augmented matrix (3.6) in Sect. 3. In Sect. 4, we analyze the backward error and forward error of the fast algorithm. Experiments are given in Sect. 5 to show the stability of the fast algorithm.

The work is partially supported by the National Key Basic Research Project 2004CB318000 of China and the Chinese National Science Foundation under Grants 10371127 and 10401035.

2. Preliminaries

We are given two polynomials $a, b \in \mathbb{R}[x]$ with $a = a_m x^m + \cdots + a_1 x + a_0$ and $b = b_n x^n + \cdots + b_1 x + b_0$, $a_m \neq 0, b_n \neq 0$. S is the Sylvester matrix of a and b. The perturbations of a and b are denoted by $\Delta a = \Delta a_m x^m + \cdots + \Delta a_1 x + \Delta a_0$ and $\Delta b = \Delta b_n x^n + \cdots + \Delta b_1 x + \Delta b_0$ respectively. We consider the minimal perturbation problem: For a positive integer $k \leq \min(m, n)$, *minimize* $\|\Delta a\|_2^2 + \|\Delta b\|_2^2$ *preserving that* $a + \Delta a$ *and* $b + \Delta b$ *have an exact GCD of degree not less than* k.

Denote $S_k = [\mathbf{a}\, A_k] \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ as the k-th Sylvester matrix,

$$S_k = \left[\begin{array}{ccccc|ccccc} a_m & 0 & \cdots & 0 & 0 & b_n & 0 & \cdots & 0 & 0 \\ a_{m-1} & a_m & \cdots & 0 & 0 & b_{n-1} & b_n & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_0 & a_1 & 0 & 0 & \cdots & b_0 & b_1 \\ 0 & 0 & \cdots & 0 & a_0 & 0 & 0 & \cdots & 0 & b_0 \end{array}\right], \qquad (2.1)$$
$$\underbrace{\qquad\qquad\qquad}_{n-k+1} \quad \underbrace{\qquad\qquad\qquad}_{m-k+1}$$

where $\mathbf{a}$ is the first column of S_k and A_k consists of the last $m+n-2k+1$ columns of S_k.

The perturbations Δa and Δb are expressed by an $(m+n+2)$-dimensional vector $\mathbf{d}$,

$$\mathbf{d} = [d_1, d_2, \ldots, d_{m+n+1}, d_{m+n+2}]^T. \qquad (2.2)$$

The k-th Sylvester structured perturbation of S_k is represented as $[\Delta\mathbf{a}\, D_k]$.

Theorem 1. [10, 11] *Given univariate polynomials* $a(x), b(x) \in \mathbb{R}[x]$ *with* $\deg(a) = m$ *and* $\deg(b) = n$. *Let* $S(a, b)$ *be the Sylvester matrix of* $a(x)$ *and* $b(x)$, S_k *be the* k*-th Sylvester matrix,* $1 \leq k \leq \min(m, n)$. *Then* $\deg(\gcd(a, b)) \geq k$ *if and only if* S_k *has rank deficiency at least* 1.

The minimal perturbation problem can be formulated as the following equality constrained least squares problem:

$$\min_{\mathbf{x},\mathbf{d}} \|\mathbf{d}\|_2, \text{ subject to } \mathbf{r} = 0, \qquad (2.3)$$

where the structured residual $\mathbf{r}$ is given by

$$\mathbf{r} = \mathbf{a} + \Delta\mathbf{a} - (A_k + D_k)\mathbf{x}. \qquad (2.4)$$

The STLN algorithm [1] initializes $\mathbf{x}$ as the unstructured least square solution $A_k\mathbf{x} \approx \mathbf{a}$ and sets $\Delta\mathbf{a} = \mathbf{d} = \mathbf{0}$, and then refines both $\mathbf{x}$ and $\mathbf{d}$ by the first order iterative update

$$\min_{\Delta\mathbf{x},\Delta\mathbf{d}} \left\| \begin{bmatrix} w(X_k - P_k) & w(A_k + D_k) \\ I_{m+n+2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{d} \\ \Delta\mathbf{x} \end{bmatrix} + \begin{bmatrix} -w\mathbf{r} \\ \mathbf{d} \end{bmatrix} \right\|_2, \qquad (2.5)$$

where w is a large penal value and I_{m+n+2} is an identity matrix of order $m+n+2$. The matrices P_k and X_k are introduced in [11, 10] such that

$$\Delta\mathbf{a} = P_k\,\mathbf{d}, \; D_k\mathbf{x} = X_k\mathbf{d}. \qquad (2.6)$$

Let us denote the coefficient matrix of the system in (2.5) by M,

$$M = \begin{bmatrix} w(X_k - P_k) & w(A_k + D_k) \\ I_{m+n+2} & \mathbf{0} \end{bmatrix}, \tag{2.7}$$

and denote $\mathbf{y} = \begin{bmatrix} \Delta\mathbf{d} \\ \Delta\mathbf{x} \end{bmatrix}$, $\mathbf{z} = \begin{bmatrix} w\mathbf{r} \\ -\mathbf{d} \end{bmatrix}$; the least squares problem (2.5) can be rewritten as

$$\min_{\mathbf{y}} \|M\mathbf{y} - \mathbf{z}\|_2 . \tag{2.8}$$

It has been shown in [11] that M is a Toeplitz-like structured matrix of displacement rank at most 4.

3. Fast Algorithm for Solving the Least Squares Problem

Fast algorithms based on QR decomposition for solving least squares problems with coefficient matrices being Toeplitz matrices have been considered in [5, 12, 4, 2, 6, 15, 17]. The stability properties of these algorithms are still not well understood and most of the algorithms may suffer from loss of accuracy when they are applied to ill-conditioned problems. Based on the method of corrected semi-normal equations, the algorithm derived in [13] can produce a more accurate R factor in the QR decomposition of a Toeplitz matrix, even for certain ill-conditioned matrices. Another fast and stable algorithm for solving the Toeplitz-like least squares problem was developed by Gu in [8]. The algorithm is based on the fast algorithm for solving Cauchy-like least squares problems. Although these algorithms [13, 8] can be used to solve least squares problems of significantly extended range from well-conditioned to certain ill-conditioned. It is still under investigation whether those algorithms can be used to solve the least squares problems (2.8) with coefficient matrices having many very small singular values. In [11], we propose to solve the least squares problem (2.8) fast by extending the fast algorithm described by Chandrasekaran et al. in [3] for solving systems of linear equations. Here, we show that their fast algorithm [3] can be generalized to solve (2.8). The numerical stability of the fast algorithm will be explained in next two sessions.

For the least squares problem (2.8), we denote its solution by $\mathbf{y}_{\text{LS}}$, and the minimum residual vector by $\mathbf{r}_{LS} = M\mathbf{y}_{\text{LS}} - \mathbf{z}$. Then $\mathbf{y}_{LS}$ solves the following linear system:

$$\begin{bmatrix} M^TM & M^T \\ M & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ -\mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z} + \mathbf{r}_{LS} \end{bmatrix}. \tag{3.1}$$

Denote by Q the orthogonal matrix from the QR decomposition of M. We partition Q as: $Q = [Q_1, Q_2]$, where $Q_2 \in \mathbb{R}^{(2m+2n-k+3)\times k}$; then

$$\|\mathbf{r}_{LS}\|_2 = \|Q_2^T\mathbf{z}\|_2 = \left\| Q_2^T \begin{bmatrix} w\mathbf{r} \\ -\mathbf{d} \end{bmatrix} \right\|_2 . \tag{3.2}$$

Due to the heavy weight of the upper block $M(1..m+n-k+1,:)$ of M, the entries of the block $Q_2(m+n-k+2..2m+2n-k+3,:)$ are $O(1)$, the block $Q_2(1..m+n-k+1,:)$ consists of near zero elements. Therefore, derived from (3.2), $\|\mathbf{r}_{LS}\|_2$ is of much smaller size compared to $\|\mathbf{z}\|_2$, i.e.

$$\|\mathbf{r}_{LS}\|_2 \ll \|\mathbf{z}\|_2 . \tag{3.3}$$

The inequality (3.3) tells us that we can compute an approximate solution $\hat{\mathbf{y}}$ to (2.8) by omitting the term $\mathbf{r}_{LS}$ and solving the following augmented system proposed in [11]:

$$\begin{bmatrix} M^TM & M^T \\ M & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ -\mathbf{z} \end{bmatrix} \approx \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix}. \tag{3.4}$$

We normalize the matrix M and the vector $\mathbf{z}$ as:

$$M := M/\|M\|_F, \quad \mathbf{z} := \mathbf{z}/\|M\|_F, \tag{3.5}$$

where $\|M\|_F$ is the Frobenius norm of M. Due to the large penal value w, after normalization, the lower left corner of M has very small diagonal elements. This causes the numerical rank deficiency of M^TM. Moreover, since M is not a square matrix, the coefficient matrix of the linear system (3.4) is rank deficient. In order to complete the generalized Schur algorithm successfully, we construct T [3, 11] as:

$$T = \begin{bmatrix} M^TM + \alpha I^{(1)} & M^T \\ M & -\beta I^{(2)} \end{bmatrix}, \tag{3.6}$$

where $\alpha I^{(1)}, \beta I^{(2)}$ are small multiples of identity matrices. Here the perturbed matrix $M^TM + \alpha I^{(1)}$ is positive definite, which ensures the positive steps complete successfully; The perturbation $\beta I^{(2)}$ is added to guarantee that the Schur complement of T with respect to $M^TM + \alpha I^{(1)}$ is negative definite, which ensures the negative steps complete successfully.

It has been shown in [11] that T is a structured matrix with displacement rank at most 10. We can construct a generator pair (G, J) for T such that

$$T - FTF^T = GJG^T,$$

where

$$F = \operatorname{diag}(Z_{m+1}, Z_{n+1}, Z_{n-k}, Z_{m-k+1}, Z_{m+n-k+1}, Z_{m+n+2}),$$

$J = \operatorname{diag}(I_4, -I_6)$, and G is a matrix with 10 columns. However, in [11], we expressed G by columns of T, for which some of entries are of order $1/w^2$, where w is the large penal value. It is undesirable for numerical stability.

In the following, we introduce a new generator matrix with columns which are only of order $1/w$, the same order as that of M. Define $t_1, \ldots, t_4$ as:

$$\begin{aligned} t_1 &= \|M(:,1)\|_2^2 + \alpha, & t_2 &= \|M(:,m+2)\|_2^2 + \alpha, \\ t_3 &= \|M(:,m+n+3)\|_2^2 + \alpha, & t_4 &= \|M(:,2n+m-k+3)\|_2^2 + \alpha. \end{aligned}$$

Let

$$\begin{aligned}
\mathbf{c}_1 &= \left[M^T(1,:)M,\, M^T(1,:)\right]^T + \alpha I(:,1),\\
\mathbf{c}_2 &= \left[M^T(m+2,:)M,\, M^T(m+2,:)\right]^T + \alpha I(:,m+2),\\
\mathbf{c}_3 &= \left[M^T(m+n+3,:)M,\, M^T(m+n+3,:)\right]^T + \alpha I(:,m+n+3),\\
\mathbf{c}_4 &= \left[M^T(2n+m-k+3,:)M,\, M^T(2n+m-k+3,:)\right]^T\\
&\quad + \alpha I(:,2n+m-k+3),
\end{aligned}$$

where I denotes the identity matrix of order $4m+4n-3k+6$. Then

$$\begin{aligned}
\mathbf{g}_1 &= \mathbf{c}_1/\sqrt{t_1},\\
\mathbf{g}_2 &= \mathbf{c}_2/\sqrt{t_2}, \text{ except that } \mathbf{g}_2[1]=0,\\
\mathbf{g}_3 &= \mathbf{c}_3/\sqrt{t_3}, \text{ except that } \mathbf{g}_3[1]=0,\ \mathbf{g}_3[m+2]=0,\\
\mathbf{g}_4 &= \mathbf{c}_4/\sqrt{t_4}, \text{ except that } \mathbf{g}_4[1]=0,\ \mathbf{g}_4[m+2]=0,\ \mathbf{g}_4[m+n+3]=0,\\
\mathbf{g}_5 &= \left[0, \mathbf{g}_1^T(2:4m+4n-3k+6)\right]^T,\\
\mathbf{g}_6 &= \left[\mathbf{g}_2^T(1:m+1), 0, \mathbf{g}_2^T(m+3:4m+4n-3k+6)\right]^T,\\
\mathbf{g}_7 &= \left[\mathbf{g}_3^T(1:m+n+2), 0, \mathbf{g}_3^T(m+n+4:4m+4n-3k+6)\right]^T,\\
\mathbf{g}_8 &= \left[\mathbf{g}_4^T(1:2n+m-k+2), 0, \mathbf{g}_4^T(2n+m-k+4:4m+4n-3k+6)\right]^T,\\
\mathbf{g}_9 &= [\ \underbrace{0,\ldots,0,0}_{2m+2n-2k+3},\ \sqrt{\beta},\ 0,\ \ldots,\ 0\ ]^T,\\
\mathbf{g}_{10} &= [\ \underbrace{0,\ldots,0,0}_{3m+3n-3k+4},\ \sqrt{\beta},\ 0,\ldots,0\ ]^T.
\end{aligned}$$

Expanding the proof in [3] by combining with singular value decomposition (SVD) of M, we prove that after applying $2m+2n-2k+3$ positive steps and $2m+2n-k+3$ negative steps of the generalized Schur algorithm, which operates on the generator pair (G,J), we can obtain a backward stable factorization of T:

$$\begin{bmatrix} \hat{R}^T & 0 \\ \hat{Q} & \hat{D} \end{bmatrix} \begin{bmatrix} \hat{R} & \hat{Q}^T \\ 0 & -\hat{D}^T \end{bmatrix}, \tag{3.7}$$

where $\hat{R}$ is upper triangular and $\hat{D}$ is lower triangular. Furthermore, using the triangular factorization (3.7) we can solve the following augmented system

$$T \begin{bmatrix} \mathbf{y} \\ \xi \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix} \tag{3.8}$$

and get

$$(T+H)\begin{bmatrix} \hat{\mathbf{y}} \\ \hat{\xi} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix}, \tag{3.9}$$

where $\|H\|_2 = O(\epsilon)$ and ϵ is the machine precision. The solution $\begin{bmatrix} \hat{\mathbf{y}} \\ \hat{\xi} \end{bmatrix}$ is obtained through the following substitutions:

$$\begin{bmatrix} \hat{R} & \check{Q}^T \\ 0 & -\hat{D}^T \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}^T & 0 \\ \hat{Q} & \hat{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix},$$

and $\hat{\mathbf{y}}$ is computed by the expression

$$\hat{R}^{-1}\hat{Q}^T\hat{D}^{-T}\hat{D}^{-1}\mathbf{z}. \tag{3.10}$$

$\hat{\mathbf{y}}$ is regarded as an approximate solution to the least squares problem (2.8).

As mentioned in [11], the computation of $\hat{\mathbf{y}}$ is of quadratic complexity $O((2m +2n-k+3)^2)$.

4. Error Analysis for the Fast Algorithm

In [3], the authors derived a backward error bound to show that the approximate solution $\hat{\mathbf{y}}$ is a backward stable solution to the original linear system. For our case, however, it is still not clear how to prove that $\hat{\mathbf{y}}$ is a backward stable solution to the least squares problem (2.8).

In the following, by means of the formula derived in [7], we compute an alternative F-norm backward error bound $\hat{\varepsilon}(\hat{\mathbf{y}})$ that $\hat{\mathbf{y}}$ satisfies. As shown by the numerical tests in Sect. 5, the obtained solutions are backward stable. Besides, based on (3.9) we derive a relative forward error bound for the approximate solution $\hat{\mathbf{y}}$ in Sect. 4.2.

4.1. Computation of Backward Error

Let $\hat{\varepsilon}(\hat{\mathbf{y}})$ be an alternative F-norm bound on δM such that $\hat{\mathbf{y}}$ is an exact solution to the least squares problem below

$$\min_{\mathbf{y}} \|(M+\delta M)\mathbf{y} - \mathbf{z}\|_2. \tag{4.1}$$

$\hat{\varepsilon}(\hat{\mathbf{y}})$ differs from the smallest possible backward perturbation $\varepsilon(\hat{\mathbf{y}})$ derived in [18, 9] by at most a factor less than 2. Suppose $M = U\begin{bmatrix} \Sigma \\ \mathbf{0} \end{bmatrix}V^T$ is the singular value decomposition of M and $\hat{\mathbf{y}} \neq \mathbf{0}$. Let

$$\hat{\mathbf{r}} = \mathbf{z} - M\hat{\mathbf{y}} = U\begin{bmatrix} \hat{\mathbf{r}}_1 \\ \hat{\mathbf{r}}_2 \end{bmatrix} \tag{4.2}$$

for $\hat{\mathbf{r}}_1 \in \mathbb{R}^{(2m+2n-2k+3)\times 1}$ and $\hat{\mathbf{r}}_2 \in \mathbb{R}^{k\times 1}$. Then

$$\hat{\varepsilon}(\hat{\mathbf{y}}) = \min(\eta, \tilde{\sigma}), \tag{4.3}$$

where $\eta = \frac{\|\hat{\mathbf{r}}\|_2}{\|\hat{\mathbf{y}}\|_2}$, and

$$\tilde{\sigma} = \sqrt{\frac{\hat{\mathbf{r}}_1^T \Sigma^2 (\Sigma^2 + \eta^2 I)^{-1} \hat{\mathbf{r}}_1}{\|\hat{\mathbf{r}}_2\|_2^2/\eta^2 + \eta^2 \hat{\mathbf{r}}_1^T (\Sigma^2 + \eta^2 I)^{-2} \hat{\mathbf{r}}_1}}. \tag{4.4}$$

The detailed analysis of computations can be found in [7].

4.2. Forward Error Analysis

We derived a relative forward error bound for the approximate solution $\hat{\mathbf{y}}$ to the least squares problem (2.8). Hereafter, we use $\kappa(\cdot)$ to denote the 2-norm condition number of its argument. We define $u = m+n-k+1, l = m+n-k+2$. For any integer $i, 1 \le i \le u+l$, σ_i is the i-th largest singular value of M.

Lemma 2. *Denote by* $\mathbf{f}$ *a vector which satisfies the following linear system:*

$$\begin{bmatrix} M^T M + \alpha I^{(1)} & M^T \\ M & -\beta I^{(2)} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{y}} \\ \hat{\xi} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix} + \mathbf{f}; \tag{4.5}$$

we derive an upper bound for $\frac{\|\hat{\mathbf{y}} - \mathbf{y}_{LS}\|_2}{\|\mathbf{y}_{LS}\|_2}$ *which is roughly of the form:*

$$\frac{\beta + \alpha\beta\kappa^2(M)}{\|M\|_2^2 - \alpha\beta\kappa^2(M)} + \frac{\kappa(M) + \beta\kappa^2(M)}{\|M\|_2^2 - \alpha\beta\kappa^2(M)} \frac{\|\mathbf{f}\|_2}{\|\mathbf{y}_{LS}\|_2}. \tag{4.6}$$

Proof. Partition $\mathbf{f}$ as $\left[\mathbf{f}_1^T, \mathbf{f}_2^T\right]^T$, $\mathbf{f}_1 \in \mathbb{R}^{(2m+2n-2k+3)\times 1}$, $\mathbf{f}_2 \in \mathbb{R}^{(2m+2n-k+3)\times 1}$; from (4.5), we have

$$\begin{cases} \left(M^T M + \alpha I^{(1)}\right) \hat{\mathbf{y}} + M^T \hat{\xi} = \mathbf{f}_1, \\ M\hat{\mathbf{y}} - \beta\hat{\xi} = \mathbf{z} + \mathbf{f}_2. \end{cases}$$

Eliminating $\hat{\xi}$, we get

$$\beta \left(M^T M + \alpha I^{(1)}\right) \hat{\mathbf{y}} + M^T M \hat{\mathbf{y}} = M^T \mathbf{z} + M^T \mathbf{f}_2 + \beta \mathbf{f}_1.$$

Noting that

$$M^T \mathbf{z} = M^T M \mathbf{y}_{LS},$$

using elementary calculus we get

$$\left(\beta M^T M + \alpha\beta I^{(1)} + M^T M\right) (\mathbf{y}_{LS} - \hat{\mathbf{y}}) = \left(\beta M^T M + \alpha\beta I^{(1)}\right) \mathbf{y}_{LS} - M^T \mathbf{f}_2 - \beta \mathbf{f}_1,$$

and

$$\begin{aligned} \mathbf{y}_{LS} - \hat{\mathbf{y}} \quad = \quad & \left(\beta I^{(1)} + \alpha\beta (M^T M)^{-1} + I^{(1)}\right)^{-1} \left(\beta I^{(1)} + \alpha\beta (M^T M)^{-1}\right) \mathbf{y}_{LS} \\ & - \left(\beta I^{(1)} + \alpha\beta (M^T M)^{-1} + I^{(1)}\right)^{-1} \left(M^\dagger \mathbf{f}_2 + \beta (M^T M)^{-1} \mathbf{f}_1\right), \end{aligned}$$

where $M^\dagger$ is the Moore-Penrose pseudoinverse of M. Since

$$\kappa(M) = \left\|M^\dagger\right\|_2 \|M\|_2, \quad \kappa^2(M) = \left\|(M^T M)^{-1}\right\|_2 \|M\|_2^2,$$

we derive an upper bound for $\frac{\|\mathbf{y}_{LS}-\hat{\mathbf{y}}\|_2}{\|\mathbf{y}_{LS}\|_2}$ which is of the form:

$$\begin{aligned} & \frac{\beta+\alpha\beta\left\|(M^TM)^{-1}\right\|_2}{1-\beta-\alpha\beta\left\|(M^TM)^{-1}\right\|_2}+\frac{\left\|M^{\dagger}\right\|_2+\beta\left\|(M^TM)^{-1}\right\|_2}{1-\beta-\alpha\beta\left\|(M^TM)^{-1}\right\|_2}\frac{\|\mathbf{f}\|_2}{\|\mathbf{y}_{LS}\|_2} \\ = & \frac{\beta\|M\|_2^2+\alpha\beta\kappa^2(M)}{(1-\beta)\|M\|_2^2-\alpha\beta\kappa^2(M)}+\frac{\kappa(M)\|M\|_2+\beta\kappa^2(M)}{(1-\beta)\|M\|_2^2-\alpha\beta\kappa^2(M)}\frac{\|\mathbf{f}\|_2}{\|\mathbf{y}_{LS}\|_2}. \end{aligned}$$

The final form of the inequality follows from $\|M\|_2 \le 1$ and $\beta \ll 1$. □

Lemma 3. *Partition* $\mathbf{z}$ *as* $\left[\mathbf{z}_u^T, \mathbf{z}_{l+k}^T\right]^T$, $\mathbf{z}_u \in \mathbb{R}^{u\times 1}$, $\mathbf{z}_{l+k} \in \mathbb{R}^{(l+k)\times 1}$; *then for* T *defined as* (3.6) *we have*

$$\left\|T^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}\end{bmatrix}\right\|_2 \le \left(\frac{4}{\sigma_u}+\frac{2}{w\beta}+\frac{1}{w\sigma_{u+l}}\right)\|\mathbf{z}_u\|_2+\left(\frac{2}{\beta}+\frac{3}{\sigma_{u+l}}\right)\|\mathbf{z}_{l+k}\|_2. \quad (4.7)$$

Proof. Denote

$$T^{-1}=\begin{bmatrix} B_{11} & B_{12}\\ B_{12}^T & B_{22}\end{bmatrix},$$

where $B_{11}\in\mathbb{R}^{(u+l)\times(u+l)}$. We derive expressions for B_{11}, B_{12} and B_{22} in terms of the SVD of M.

Let $M = U\begin{bmatrix}\Sigma\\ \mathbf{0}_{k\times(u+l)}\end{bmatrix}V^T$ be the SVD of M, where U, V are orthogonal matrices and Σ is a diagonal matrix consisting of singular values of M. The diagonal matrix Σ can be written as

$$\Sigma=\begin{bmatrix}\Sigma_u & \\ & \Sigma_l\end{bmatrix},$$

where

$$\Sigma_u=\operatorname{diag}(\sigma_1,\ldots,\sigma_u),\ \Sigma_l=\operatorname{diag}(\sigma_{u+1},\ldots,\sigma_{u+l}).$$

We define diagonal matrices

$$\begin{aligned} \Lambda_u &= \left(\left[I_u+\alpha\Sigma_u^{-2}\right]^{-1}+\beta I_u\right)^{-1}, \\ \Lambda_u' &= \left(\Sigma_u+\alpha\Sigma_u^{-1}\right)^{-1}\Lambda_u, \\ \Lambda_l &= \left(\Sigma_l+\alpha\Sigma_l^{-1}\right)^{-1}\left(\left[I_l+\alpha\Sigma_l^{-2}\right]^{-1}+\beta I_l\right)^{-1}, \\ \Lambda_{l+k} &= \begin{bmatrix}\left(\left[I_l+\alpha\Sigma_l^{-2}\right]^{-1}+\beta I_l\right)^{-1} & \\ & \frac{1}{\beta}I_k\end{bmatrix}, \end{aligned}$$

where I_u, I_l, I_k denote identity matrices of dimensions u, l, k respectively. Then

$$B_{11} = V \begin{bmatrix} \beta\Sigma_u^{-1}\Lambda'_u & \\ & \beta\Sigma_l^{-1}\Lambda_l \end{bmatrix} V^T, \tag{4.8}$$

$$B_{12} = V \begin{bmatrix} \Lambda'_u & & \mathbf{0}_{u\times k} \\ & \Lambda_l & \mathbf{0}_{l\times k} \end{bmatrix} U^T, \tag{4.9}$$

$$B_{22} = -U \begin{bmatrix} \Lambda_u & \\ & \Lambda_{l+k} \end{bmatrix} U^T. \tag{4.10}$$

Note that the following inequalities hold:

$$\|\Lambda_u\|_2 \le 1 + 1/\sigma_u,\ \|\Lambda'_u\|_2 \le 1/\sigma_u,\ \|\Lambda_l\|_2 \le 1/\sigma_{u+l},\ \|\Lambda_{l+k}\|_2 \le 1 + 1/\beta; \tag{4.11}$$

for the last inequality the following assumption is used:

$$\alpha\beta < \sigma_{u+l}^2. \tag{4.12}$$

Meantime, for the partition of U:

$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}, \text{ where } U_{11} \in \mathbb{R}^{u\times u},$$

due to the heavy weight ($O(w)$) of the submatrix $M(1..u, :)$, the following inequalities hold:

$$\|U_{12}\|_2,\ \|U_{21}\|_2 \le 1/w. \tag{4.13}$$

Finally, we derive that

$$\begin{aligned} \left\| T^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{z} \end{bmatrix} \right\|_2 &\le \|B_{12}\mathbf{z}\|_2 + \|B_{22}\mathbf{z}\|_2 \\ &\le \left(\frac{4}{\sigma_u} + \frac{2}{w\beta} + \frac{1}{w\sigma_{u+l}} \right) \|\mathbf{z}_u\|_2 + \left(\frac{2}{\beta} + \frac{3}{\sigma_{u+l}} \right) \|\mathbf{z}_{l+k}\|_2 . \end{aligned}$$

□

Lemma 4. *Assume that H is a backward error matrix which satisfies* (3.9), *and that $\gamma\|H\|_2 < 1$, where*

$$\gamma = \beta/\sigma_{u+l}^2 + 2/\sigma_{u+l} + 1/\beta + 1; \tag{4.14}$$

then for the vector $\mathbf{f}$ *defined in Lemma* 2, *we have*

$$\|\mathbf{f}\|_2 \le \frac{\|H\|_2}{1 - \gamma\|H\|_2} \left[\left(\frac{4}{\sigma_u} + \frac{2}{w\beta} + \frac{1}{w\sigma_{u+l}} \right) \|\mathbf{z}_u\|_2 + \left(\frac{2}{\beta} + \frac{3}{\sigma_{u+l}} \right) \|\mathbf{z}_{l+k}\|_2 \right]. \tag{4.15}$$

Proof. By the definitions of $\mathbf{f}$ and H, we get

$$\begin{aligned}\mathbf{f} &= -H\begin{bmatrix}\hat{\mathbf{y}}\\ \hat{\xi}\end{bmatrix}\\ &= -H\left(T+H\right)^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}\end{bmatrix}\\ &= -H\left(I+T^{-1}H\right)^{-1}T^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}\end{bmatrix}.\end{aligned}$$

So

$$\begin{aligned}\|\mathbf{f}\|_2 &\le \frac{\|H\|_2}{1-\|T^{-1}H\|_2}\left\|T^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}\end{bmatrix}\right\|_2\\ &\le \frac{\|H\|_2}{1-\|T^{-1}\|_2\,\|H\|_2}\left\|T^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}\end{bmatrix}\right\|_2.\end{aligned}$$

We use (4.8), (4.9), (4.10) and (4.11) to derive that

$$\begin{aligned}\left\|T^{-1}\right\|_2 &\le \|B_{11}\|_2+\|B_{12}\|_2+\|B_{22}\|_2\\ &\le \beta/\sigma_{u+l}^2+2/\sigma_{u+l}+1/\beta+1.\end{aligned}$$

Based on Lemma 3 and the assumption of $\gamma\|H\|_2<1$, we get the final form of the inequality. □

Theorem 5. *Assume that $\hat{\mathbf{y}}$ is a vector which solves the linear system* (3.9)*; with assumptions of $\alpha\beta<\sigma_{u+l}^2$ and $\gamma\|H\|_2<1$, we can derive an upper bound for the relative forward error $\frac{\|\hat{\mathbf{y}}-\mathbf{y}_{LS}\|_2}{\|\mathbf{y}_{LS}\|_2}$ which is of the form:*

$$\frac{\beta+\alpha\beta\kappa^2(M)}{\|M\|_2^2-\alpha\beta\kappa^2(M)}+\frac{\kappa(M)+\beta\kappa^2(M)}{\|M\|_2^2-\alpha\beta\kappa^2(M)}\frac{\|H\|_2\,\delta}{(1-\gamma\|H\|_2)\|\mathbf{y}_{LS}\|_2}, \tag{4.16}$$

where γ is defined in (4.14) *and*

$$\delta=\left(\frac{4}{\sigma_u}+\frac{2}{w\beta}+\frac{1}{w\sigma_{u+l}}\right)\|\mathbf{z}_u\|_2+\left(\frac{2}{\beta}+\frac{3}{\sigma_{u+l}}\right)\|\mathbf{z}_{l+k}\|_2\,. \tag{4.17}$$

Proof. It follows immediately from Lemmas 2, 3 and 4. □

Remark 4.1. In order to complete the generalized Schur algorithm stably we take α,β that satisfy the lower bound derived in [3]. In essential, it means that α,β can be taken somewhat larger than the machine precision. Besides this, based on the error analysis in this section, α,β should be taken such that the upper bound (4.16) is as small as it could be. Hence the assumption $\alpha\beta<\sigma_{u+l}^2$ used in Theorem 5 is natural, noting that it is approximately equivalent to $\alpha\beta\kappa^2(M)<1$, which is a necessary condition that the upper bound (4.16) is smaller than 1.

Remark 4.2. In practice, after normalization (3.5), the following inequalities hold:

$$\|\mathbf{z}_u\|_2<1,\quad \|\mathbf{z}_{l+k}\|_2<1/w. \tag{4.18}$$

Furthermore, when the given integer k (2.5) is taken as an upper bound of the degree of approximate GCD, we generally have $\delta < 1$.

5. Experiments

In Table 1, we show the performance of the fast version of the algorithm AppSylv-k [10]. The efficiency is gained by applying the fast algorithm described in Sect. 3 to solve the least squares problem (2.8) in each iteration. In the numerical tests, we compute minimal perturbations needed for two univariate polynomials having an exact GCD of degree not less than a given integer. All computations are done in Maple 10 under Windows for Digits $= 15$, $\alpha = \beta = 10^{-14}$.

Table 1. Algorithm performance on benchmarks

Ex	m,n	k	error (classic)	error (new fast)	for.err. (LS Prob.)	$\hat{\varepsilon}(\hat{\mathbf{y}})$ (LS Prob.)
1	$2, 2$	1	5.59933e–3	5.59933e–3	0.461e–3	0.238e–12
2	$3, 3$	2	1.07129e–2	1.07129e–2	0.434e–3	0.176e–13
3	$5, 4$	3	1.56146e–6	1.56146e–6	0.143e–2	0.143e–13
4	$5, 5$	3	1.34138e–8	1.34318e–8	0.664e–3	0.452e–13
5	$6, 6$	4	1.96333e–10	1.96333e–10	0.182e–3	0.448e–13
6	$8, 7$	4	1.98415e–16	1.98416e–16	0.322e–3	0.896e–14
7	$10, 10$	5	1.51551e–12	1.51552e–12	0.272e–2	0.598e–13
8	$14, 13$	7	2.61818e–4	2.61819e–4	0.163e–1	0.112e–12
9	$28, 28$	10	2.54575e–4	3.54600e–4	0.992e–1	0.512e–14
10	$50, 50$	30	9.35252e–6	9.40237e–6	0.134	0.168e–13

The sample polynomials are the same as those generated in [10]: For each example, we use 50 random cases for each (m, n), and report the average over all results. For each example, the prime parts and GCD of two polynomials are constructed by choosing polynomials with random integer coefficients in the range $-10 \le c \le 10$, and then adding a perturbation. For noise we choose a relative tolerance 10^{-e}, then randomly choose a polynomial that has the same degree as the product, with coefficients in $[-10^e, 10^e]$. Finally, we scale the perturbation so that the relative error is 10^{-e}.

In Table 1, m, n denote the degrees of polynomials a and b; k is a given integer; "error (classic)" and "error (new fast)" denote the minimal perturbations computed by the algorithms given in [10] and this report respectively; "for.err. (LS Prob.)" denotes the relative forward error of $\hat{\mathbf{y}}$ with respect to the solution given in [10]. In the last column we show backward perturbations $\hat{\varepsilon}(\hat{\mathbf{y}})$ to the least squares problem (2.8) computed according to method given in Sect. 4.1.

The small backward perturbation $\hat{\varepsilon}(\hat{\mathbf{y}})$ shown in Table 1 imply that $\hat{\mathbf{y}}$ is a stable solution to (2.8). The computed minimal polynomial perturbations have the same magnitudes as those computed by the algorithm in [10]; Especially, the new results in the first eight examples even have several significant digits identical to that of the classic results [10]. However, from the last two examples, we can see that the accuracy of the new results could be affected by the large condition number of M.

Acknowledgment
The authors would like to thank Prof. Zhongzhi Bai for useful discussions.

References

[1] A.A. Anda and H. Park, *Fast plane with dynamic scaling*, SIAM J. Matrix Anal. Appl. **15**, pp. 162–174, 1994.

[2] A.W. Bojanczyk, R.P. Brent and F.R. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math. **49**, pp. 81–94, 1986.

[3] S. Chandrasekaran and A.H. Sayed, *A fast stable solver for nonsymmetric Toeplitz and quasi-Toeplitz systems of linear equations*, SIMAX **19**, pp. 107–139, 1998.

[4] J. Chun, T. Kailath and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput. **8**, pp. 899–913, 1987.

[5] G. Cybenko, *A general orthogonalization technique with applications to time series analysis and signal processing*, Math. Comp. **40**, pp. 323–336, 1983.

[6] G. Cybenko, *Fast Toeplitz orthogonalization using inner products*, SIAM J. Sci. Stat. Comput. **8**, pp. 734–740, 1987.

[7] M. Gu, *Backward perturbation bounds for linear least squares problems*, SIAM J. Matrix Anal. Appl. **20**, pp. 363–372, 1998.

[8] M. Gu, *New fast algorithms for structured linear least squares problems*, SIAM J. Matrix Anal. Appl. **20**, pp. 244–269, 1998.

[9] N.J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[10] E. Kaltofen, Z. Yang and L. Zhi, *Structured low rank approximation of a Sylvester matrix.* In Dongming Wang and Lihong Zhi, editors, International Workshop on Symbolic-Numeric Computation Proceedings, pp. 188–201, 2005.

[11] B. Li, Z. Yang and L. Zhi, *Fast low rank approximation of a Sylvester matrix by structured total least norm*, Journal of Japan Society for Symbolic and Algebraic Computation **11**, pp. 165–174, 2005.

[12] J.G. Nagy, *Fast inverse QR factorization for Toeplitz matrices*, SIAM J. Sci. Stat. Comput. **14**, pp. 1174–1183, 1993.

[13] H. Park and L. Eldén, *Stability analysis and fast algorithms for triangularization of Toeplitz matrices*, Numer. Math. **76**, pp. 383–402, 1997.

[14] H. Park, L. Zhang and J.B. Rosen, *Low rank approximation of a Hankel matrix by structured total least norm*, BIT **39**, pp. 757–779, 1999.

[15] S. Qiao, *Hybrid algorithm for fast Toeplitz orthogonalization*, Numer. Math. **53**, pp. 351–366, 1988.

[16] J.B. Rosen, H. Park and J. Glick, *Total least norm formulation and solution for structured problems*, SIAM J. Matrix Anal. Appl. **17**, pp. 110–128, 1996.

[17] D.R. Sweet, *Fast Toeplitz orthogonalization*, Numer. Math. **43**, pp. 1–21, 1984.

[18] B. Waldén, R. Karlson and J.-G. Sun, *Optimal backward perturbation bounds for the linear least square problem*, Numer. Lin. Alg. Appl. **2**, pp. 271–286, 1995.

Bingyu Li, Zhuojun Liu and Lihong Zhi
Key Laboratory of Mathematics Mechanization
AMSS, Chinese Academy of Sciences
Beijing 100080 China
e-mail: {liby,zliu,lzhi}@mmrc.iss.ac.cn

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 99–112
© 2007 Birkhäuser Verlag Basel/Switzerland

New Algorithms for Exact and Approximate Polynomial Decomposition

Mark Giesbrecht and John May

Abstract. Computing a decomposition of a polynomial $f(x)$ as a functional composition $g(h(x))$ of polynomials $g(x)$ and $h(x)$ is an important and well-studied problem, both for exact and approximate inputs. In this paper, we re-examine the original (exponential-time) algorithm of Barton and Zippel for this task, which looks for special factors of an associated *separated* bivariate polynomial. We demonstrate algorithms using this approach which are reasonably fast (i.e., run in a polynomial number of operations) for exact computation, and provide an effective new approach for the decomposition of approximate polynomials. For approximate polynomials we exhibit rigorous lower bounds on the distance to the nearest decomposable polynomial, as well as robust numerical algorithms.

1. Introduction

Given a polynomial $f \in \mathsf{K}[x]$ of degree n over a field K, the problem of polynomial decomposition asks if there exist polynomials $g, h \in \mathsf{K}[x]$ such that $f(x) = g(h(x))$ with $1 < \deg g, \deg h < n$. This problem has been studied for exact polynomials and rational functions by many authors, including Alonso, Gutiérrez, and Recio (1995), Gutiérrez, Recio, and de Velasco (1988), Kozen and Landau (1989), von zur Gathen (1990), and Zippel (1991).

In Corless, Giesbrecht, Jeffrey, and Watt (1999), the problem of the functional decomposition of approximate polynomials is examined. That is, for a given polynomial $f \in \mathbb{C}[x]$, polynomials $g, h \in \mathbb{C}[x]$ are sought such that there exists a "small" $f_\triangle$ with $f(x)+f_\triangle(x) = g(h(x))$. Here "small" is measured by the coefficient 2-norm: for

$$u = \sum_{0\le i\le m} u_i x^i \in \mathbb{C}[x], \text{ the coefficient 2-norm is defined by } \|u\|^2 = \sum_{0\le i\le m} u_i\overline{u_i},$$

where $\overline{u_i}$ is the complex conjugate of u_i.

In Sect. 1 of this paper we reconsider the original algorithm of Barton and Zippel (1976) for decomposing a polynomial $f \in \mathbb{C}[x]$. Their algorithm is based upon the fact that, for any right composition factor h of f, $h(x) - h(y)$ divides $f(x) - f(y)$. We show that, except for a particular easily handled class of polynomials, their algorithm in fact runs in polynomial time. In Sect. 2 we exhibit a reduction from the problem of decomposition to that of (structured) bivariate factoring to look at approximate polynomial decomposition. We demonstrate algorithms to provide rigorous lower bounds on the distance to an indecomposable polynomials (using the techniques of Kaltofen and May (2003)). In Sect. 3, we show how to use recent approximate bivariate factoring algorithms for polynomials to compute approximate decompositions of polynomials. Finally, in Sect. 4, we perform an empirical analysis on our new algorithm as compared to the techniques of Corless et al. (1999).

2. Barton and Zippel's Algorithm Revisited

Given a polynomial $f \in \mathsf{K}[x]$, over a field K, we first consider the problem of determining if f can be functionally decomposed in $\mathsf{K}[x]$, that is, determining if there exist $g, h \in \mathsf{K}[x]$, of degrees r, $s \geq 2$ respectively (with $n = r \cdot s$), such that $f(x) = g(h(x)) = (g \circ h)(x)$. The polynomial h is called a right composition factor of f. If f does decompose, we compute a decomposition.

The first known algorithm to compute the functional decomposition of a polynomial was given in Barton and Zippel (1976), and relied upon the following theorem of Fried and MacRae:

Fact 2.1 (Fried and MacRae, 1969). *Let x, y be independent indeterminates over a field K and $f, h \in \mathsf{K}[x]$. Then $h(x) - h(y)$ divides $\Phi_f = (f(x) - f(y))/(x - y)$ if and only if $f(x) = g(h(x))$ for some $g \in \mathsf{K}[x]$.*

Polynomials of the form Φ_f are called separated polynomials, and for any polynomial $h \in \mathsf{K}[x]$, we write Φ_h for $(h(x) - h(y))/(x - y)$. The above fact leads directly to the decomposition algorithm from Barton and Zippel (1976, 1985):

Algorithm 2.2.

```
    INPUT: A polynomial $f \in \mathsf{K}[x]$.
    OUTPUT: $g, h \in \mathsf{K}$ such that $f = g \circ h$, or $f$ is indecomposable.
1. Form $\Phi_f = (f(x) - f(y))/(x - y)$;
2. Factor $\Phi_f$ completely over $\mathbb{C}[x, y]$; if $\Phi_f$ is irreducible, $f$ is indecomposable;
3. Examine all factors of $\Phi_f$, looking for factors of the form $\Phi_h = ((h(x) - h(y))/(x - y)$ for some $h \in \mathsf{K}[x]$;
4. If no factors of the form $\Phi_h$ exist then $f$ is indecomposable, otherwise, we
   have found a factor $h$ from which we can compute $g$.
```

Note that in step 4, computing g from a given h is simply a matter of solving the system of equations

$$f - \sum_{i=0}^{r} g_i\, h^i = 0, \tag{2.1}$$

which are linear in $g_0, g_1, \ldots, g_r \in \mathsf{K}$, where $g = \sum_{0 \le i \le r} g_i x^i$, and $r = \deg g = n/\deg h$.

The running times of steps 1, 2, and 4 of Algorithm 2.2 are clearly polynomial in the degree of f. However, the possibility of having to check what could be exponentially many combinations of factors of Φ_f in step 3 prevents the algorithm from having a running time which is polynomial in the degree.

We can avoid this exponential-time step, at least for *tame* polynomials as follows. All $f \in \mathsf{K}[x]$ are tame when the characteristic of K is zero. When the characteristic char K of K is $p > 0$, then a polynomial is tame if $p > \deg f$ (a more inclusive definition of tame polynomials is given in (Turnwald, 1995, Definition 4.1) and von zur Gathen (1990)). Of course, Barton and Zippel's (1985) algorithm works over any field, whereas we confine ourselves to the tame case in the remainder of this paper.

The following theorem of Turnwald (1995) is a refinement of Fried's (1970) breakthrough solution of "Schur's conjecture":

Fact 2.3 (Turnwald, 1995). *Let $f \in \mathsf{K}[x]$ be tame and indecomposable of degree $n > 1$. Suppose that n is not an odd prime and it is not the case that $f(x) = \alpha D_n(a, x+b)+\beta$ for $\alpha, \beta, a, b \in \mathsf{K}$, where $a = 0$ if $n = 3$. If $f(x)$ is indecomposable, then $(f(x) - f(y))/(x - y)$ is absolutely irreducible.*

The notation $D_n(a, x)$ refers to a Dickson polynomial. The Dickson polynomials can be defined as the compositions of Chebyshev polynomials, linear polynomials, and polynomials of the form x^m, or more concretely as

$$\begin{aligned} D_n(x, a) = &\sum_{i=0}^{\lfloor n/2 \rfloor} \frac{n}{n-i} \binom{n-i}{i} (-a)^i x^{n-2i} \\ =& x^n - nax^{n-2} + n(n-3)/2 \cdot a^2 x^{n-4} + \cdots \end{aligned}$$

The implications of Fact 2.3 for decomposition are that if f is not of prime degree and Φ_f factors in $\mathsf{K}[x, y]$ then f decomposes. Also, if h is an indecomposable composition factor of f then Φ_h is irreducible in $\mathsf{K}[x, y]$ unless h is a Dickson polynomial. Thus, searching combinations of factors of Φ_f is not necessary unless f has only Dickson polynomials as right composition factors. We show that this latter case is not difficult to handle, as it turns out that we can detect if a polynomial f has Dickson polynomials as right composition factors simply by examining the three highest-order coefficients of f.

Lemma 2.4. *Suppose $f \in \mathsf{K}[x]$ is monic and has a right composition factor of prime degree $q \geq 3$ which is a Dickson polynomial $D_q(a, x+b)$. Then f has the form:*

$$f = x^n + nb\,x^{n-1} + \left(\frac{n(n-1)}{2}\,b^2 - n\,a\right) x^{n-2} + \cdots$$

Proof. If $f \in \mathsf{K}[x]$ is monic and $f = g \circ D_q(a, x+b)$ then g is monic as well (since D_q is monic). If $q \geq 3$ is prime, then

$$D_q(a, x+b) = x^q + q\,b\,x^{q-1} + \left(\frac{q(q-1)}{2} b^2 - q\,a\right) x^{q-2} + \cdots,$$

so

$$\begin{aligned} g(D_q(a, x+b)) &= \left(x^q + q\,b\,x^{q-1} + \left(\frac{q(q-1)}{2} b^2 - q\,a\right) x^{q-2} + \cdots\right)^k + \cdots \\ &= x^{q\,k} + k\,(q\,b\,x^{q-1})\,(x^q)^{k-1} + \frac{k(k-1)}{2}\,(q\,b\,x^{q-1})^2\,(x^q)^{k-2} \\ &\qquad + k\left(\frac{q(q-1)}{2} b^2 - q\,a\right) x^{q-2}\,(x^q)^{k-1} + \cdots \\ &= x^n + nb\,x^{n-1} + \left(\frac{n(n-1)}{2}\,b^2 - n\,a\right) x^{n-2} + \cdots \end{aligned}$$

□

This yields the following improved version of Barton and Zippel's algorithm.

Algorithm 2.5.

INPUT: A tame polynomial $f = x^n + f_{n-1}\,x^{n-1} + f_{n-2}\,x^{n-2} + \cdots \in \mathsf{K}[x]$.
OUTPUT: $g, h \in \mathsf{K}$ such that $f = g \circ h$, or a message that f is indecomposable.

1. Form $\Phi_f = (f(x) - f(y))/(x - y)$;
2. Factor Φ_f completely in $\mathsf{K}[x, y]$; if Φ_f is irreducible, f is indecomposable;
3. Attempt to find an irreducible factor of Φ_f of the form Φ_h for some $h \in \mathsf{K}[x]$;
4. If such an h was found in step 3, compute g from the system (2.1) so that $f(x) = g(h(x))$;
5. If no such h was found in step 3, for each prime number q which divides $n = \deg f$, determine if $h(x) = D_q(a, x+b)$ is a right composition factor of f, where

$$b = f_{n-1}/n, \;\; a = \frac{1}{n}\left(\frac{n(n-1)}{2}\,(f_{n-1}/n)^2 - f_{n-2}\right);$$

   This can be done by attempting to compute a left composition factor g by solving the system (2.1), and testing that $f(x) = g(h(x))$.

Algorithm 2.5 works for similar reasons to Barton and Zippel's, except now we know that the Φ_h corresponding to a non-Dickson factor must be irreducible by Fact 2.3 (and Dickson factors are dealt with in step 5). It clearly requires only a polynomial number of operations since the factor combination of Algorithm 2.2 has been eliminated and step 5 involves trying fewer than n possibilities for q.

While Algorithm 2.5 is interesting as a theoretical and historical artifact, in that a slight modification of the first algorithm of Barton and Zippel in fact runs in polynomial-time, it is really not competitive for exact decomposition of polynomials. A faster polynomial time algorithm from Kozen and Landau (1989), and a nearly linear time algorithm in von zur Gathen (1990), have been known for over 15 years. However, Algorithm 2.5 lends itself well to approximate decomposition, as we shall see in the next section.

3. Approximate Decomposability Testing

The relationship between decomposition and bivariate factorization presented in the previous section make it straightforward to transform decomposition into a linear problem. This provides a useful approach to approximate decomposition using recent approximate bivariate decomposition and irreducibility testing algorithms.

For a given $f = \sum_{i=0}^{n} f_i\, x^i \in \mathbb{R}[x]$ (with n not prime), we know f decomposes if and only if

$$\Phi_f = \frac{f(x) - f(y)}{x - y} = \sum_{i=1}^{n} f_i \left(\sum_{j=0}^{i-1} x^{i-j-1} y^j \right) \tag{3.1}$$

factors in $\mathbb{C}[x, y]$. Kaltofen and May (2003) provide a method for computing a lower bound on how far an irreducible bivariate polynomial is from a reducible polynomial. Their method employs (and extends) the linearization of Ruppert (1999). Stated in a somewhat non-standard way, Ruppert shows that for any n, m positive integers, there exist matrices $R_{ij} \in \mathbb{Z}^{4mn \times 2mn+n-1}$ for $0 \le i < m$ and $0 \le j < n$, with the following properties. Let

$$w = \sum_{0 \le i \le n} \sum_{0 \le j \le m} w_{ij} x^i y^j \in \mathbb{R}[x, y].$$

Then

$$\text{Rup}(w) = \sum_{0 \le i \le n} \sum_{0 \le j \le m} w_{ij} R_{ij} \in \mathbb{R}^{4mn \times 2mn+n-1} \tag{3.2}$$

has full rank if and only if w is absolutely irreducible. Kaltofen and May (2003) show that if w is irreducible and $\tilde{w}$ does not have degree greater than w in either variable, then

$$\|w - \tilde{w}\|_2 < \frac{\sigma(\text{Rup}(w))}{\max\{m, n\}\sqrt{2mn - n}}$$

implies that $\tilde{w}$ is irreducible, where $\sigma(\text{Rup}(w))$ is the smallest singular value of $\text{Rup}(w)$. They also show that $\|R_{ij}\| < \max\{n, m\}$. Note that Kaltofen and May (2003)'s result is stronger than an immediate application of Ruppert's theorem as it allows the degree of $\tilde{w}$ to be smaller than that of w.

Returning to the decomposition problem, Φ_f has factors if and only if the matrix $\text{Rup}(\Phi_f)$ does not have full rank. Thus, we can bound the distance of an indecomposable f to a decomposable polynomial by bounding the distance of

$\text{Rup}(\Phi_f)$ to a matrix of lower rank, as done in Kaltofen and May (2003) through the singular value decomposition.

Theorem 3.1. *If $f \in \mathbb{R}[x]$ is an indecomposable polynomial, and $\tilde{f} \in \mathbb{R}[x]$ is a decomposable polynomial with $\tilde{f}(0) = f(0)$ and $\deg \tilde{f} \leq \deg f$ then*

$$\|f - \tilde{f}\|_2 \geq \frac{\sigma(\text{Rup}(\Phi_f))}{n^2\sqrt{2n^2 - n}}.$$

Proof. Since Φ_f is irreducible and $\Phi_{\tilde{f}}$ is not, and $\deg_x \Phi_f = \deg_y \Phi_f = d$ we have, from Theorem 1 of Kaltofen and May (2003),

$$\|\Phi_f - \Phi_{\tilde{f}}\|_2 \geq \frac{\sigma(\text{Rup}(\Phi_f))}{n\sqrt{2n^2 - n}}.$$

Looking at (3.1) it is easy to see that:

$$\|\Phi_f - \Phi_{\tilde{f}}\|_2 = \|\Phi_{f-\tilde{f}}\|_2 \leq n\,\|(f - \tilde{f}) - (f - \tilde{f} \bmod x)\|_2 \leq n\,\|f - \tilde{f}\|_2.$$

Thus

$$\|f - \tilde{f}\|_2 \geq \frac{\sigma(\text{Rup}(\Phi_f))}{n^2\sqrt{2n^2 - n}}.$$

□

We now have the ability to compute a radius of indecomposability about any indecomposable polynomial and, as with irreducibility, this gives a simple algorithm to test the indecomposability of an approximate polynomial when a tolerance on the coefficients is specified.

Note that, as with factorization, if we omit the degree bound, it is possible to find a decomposable polynomial which is arbitrarily close to f, namely $f \circ (\epsilon x^2 + x)$. It may be that the degree bound in Theorem 3.1) is too tight; approximate decompositions up to degree $2 \deg f - 1$ may be meaningful. However, we will consider only approximate decompositions of the same degree.

Example 3.2. We begin with a decomposable monic polynomial with a large noise term added to it (making it indecomposable):

$$f = (x^2 - x) \circ (x^2 + 3\,x) + .02\,x^3 = x^4 + 6.02\,x^3 + 8\,x^2 - 3\,x.$$

Then we compute

$$\Phi_f = x^3 + x^2y + xy^2 + y^3 + 6.02\,(x^2 + xy + y^2) + 8.0\,(x + y) - 3.0,$$

and the matrix $\text{Rup}(\Phi_f)$ which is 27×20. Computing the largest coefficient of $\|\text{Rup}(\Phi)\|^2$ (where Φ is a polynomial with symbolic coefficients of the same form as Φ_f) we get 200 (compared to the bound $d^2\,(2d^2 - d) = 7168$ from Theorem 3.1). Computing the smallest singular value of $\text{Rup}(\Phi_f)$, we find a lower bound on the distance from f to the nearest polynomial which decomposes (in the 2-norm) is 4.32207×10^{-5}. Thus, any polynomial with constant coefficient 0 which is closer than the bound must also be indecomposable.

If we want to compute a better bound to separate f from decomposable monic polynomials, we can consider the largest coefficient of $\|\text{Rup}(\Phi)\|^2$ after substituting

0 for the symbol corresponding to coefficient of the highest total degree terms. In that case we get 100 which leads to a slightly larger bound of 5.10581×10^{-5}.

4. Approximate Decomposition

In this section we describe how to use the established connection between the decomposability of f and the irreducibility of Φ_f as a basis for decomposition algorithms. For many polynomials it is still probably best to use one of the algorithms described in Corless, Giesbrecht, Jeffrey, and Watt (1999). However, if those algorithms perform badly, we would like an alternate approach, which in some sense has a firmer theoretical underpinning. Ultimately neither of these algorithms provides a guaranteed solution in all cases.

Approximate Decomposition Using Approximate Bivariate Factorization

We can build an algorithm to compute approximate decompositions of polynomials using the reduction to approximate bivariate polynomial factorization. The straightforward application of the approximate factoring algorithm described in Gao, Kaltofen, May, Yang, and Zhi (2004) to Φ_f creates an approximate version of Algorithm 2.5, which we explore now.

We first note that an algorithm which finds the nearest separated reducible polynomial $\Phi_{\tilde{f}} \in \mathbb{R}[x,y]$, of the form $(\tilde{f}(x) - \tilde{f}(y))/(x-y)$, for some $\tilde{f} \in \mathbb{R}[x]$, would actually find the nearest decomposable polynomial $\tilde{f}$ to f. However, this is the problem of looking for nearby reducible *structured* (i.e., separated) polynomials, which we will address in the next subsection. In this subsection we work to reconstruct structured factors from the factorization of a nearby unstructured polynomial.

Algorithm 4.1.

```
INPUT: An indecomposable polynomial $f \in \mathbb{R}[x]$ of degree $n$.
OUTPUT: $g, h \in \mathbb{R}[x]$ such that $f \approx g \circ h$.
1. Form $\Phi_f = (f(x) - f(y))/(x-y)$;
2. Compute an approximate factorization of $\Phi_f$ over $\mathbb{C}[x,y]$; discard all factors
   $p$ such that $\operatorname{tdeg} p + 1$ does not divide $\deg f$ – if no factors remain, skip to
   step 5;
3. For each remaining factor compute its distance to a factor of the form $\Phi_h$;
  4.1 For all terms of the same total degree, compute the standard deviation
      of their coefficients;
  4.2 Find the maximum of the deviations over all sets of terms, and set this
      as the distance to a separated factor (see (3.1));
5. Choose the factor $\Upsilon$ of $\Phi$ with the smallest distance to a separated polyno-
   mial, and form $h = \sum_{i=1}^{n} f_i x^i$ where $f_i$ is the average of the coefficients of the
   terms of $\Upsilon$ with total degree $i-1$; use $h$ to compute a least squares solution
   $g$ to the system (2.1), so that $\|f - g \circ h\|$ is minimized;
```

6. Find a and b as in Algorithm 2.5 step 5, and compute a corresponding g by least squares for each possible choice of q;
7. Improve each approximate decomposition with Gauss-Newton iteration as in Corless et al. (1999);
8. Return the g, h pair with the smallest value of $\|f - g \circ h\|_2$.

In practice, if f is a slightly perturbed decomposable polynomial then the approximate factors of Φ_f tend to contain polynomials very close to the form Φ_h. However, there is no guarantee on how close the approximate factors will come to having this form.

If one finds that an approximate factorization of Φ_f does not have any factors of the correct degree, it is possible to modify the approximate GCD algorithm used in the approximate factorization to produce factors of a predetermined degree (by choosing what the numerical rank of the Sylvester matrix will be, rather than trying to compute what it should be). In this way one can always find an approximate decomposition without a Dickson polynomial as a right decomposition factor though the backward error may be quite bad in some cases.

In step 6 we suggest a simple least squares heuristic to compute the nearest polynomial with a right Dickson factor. In fact, since the Dickson polynomials are defined by only two parameters ($a, b \in \mathbb{R}$), it is easy to show that the absolutely nearest polynomial to f with a Dickson right factor can be found by minimizing a bivariate rational function of degree $O((\deg f)^4)$. This can be solved in polynomial time in $\deg f$ and $\log \|f\|$ by exact methods (see, e.g., Renegar 1992). While we make no claim that this method is at all practical, it is interesting to note that it exists.

In the following example, step 6 of the algorithm is omitted.

Example 4.2. Beginning with the same polynomial as in Example 3.2

$$f = (x^2 - x) \circ (x^2 + 3\,x) + .02\,x^3,$$

we feed Φ_f into an approximate factorization algorithm and get the following factorization:

$$\begin{aligned}\Phi_f \approx{} & (4.9047250 + 1.6439030\,x + 1.6439030\,y) \\ & \cdot (-0.61172559 + 1.836216\,x + 1.836216\,y + 0.6115972\,x^2 \\ & \qquad - 0.003454426\,xy + 0.6115972\,y^2)\end{aligned}$$

The first factor is closer to the form Φ_h than the second, and the best fit h is

$$h(x) = 3.00099703989216\,x + x^2,$$

which we made monic to give a neater decomposition. Using least squares to solve for the best corresponding monic g, we get

$$g = -1.00029893310899\,x + x^2.$$

Composing, we obtain

$g(h(x)) = -3.00189413726736\,x + 8.00568430033252\,x^2 + 6.00199407978432\,x^3 + x^4,$

which has distance 0.0189766 from the original f.

As with factoring, if the smallest singular value of $\mathrm{Rup}(\Phi_f)$ is quite large then it may be trivial to find a closer decomposition than the one produced by the algorithm. Most trivial approximate decompositions have relative backward error of about 1. For example, setting the coefficients of all the odd power terms to 0 will given a polynomial which has a right composition factor of x^2. For a randomly generated polynomial, not close to one that decomposes, this may be the best we can do.

Approximate Decomposition by Searching the Ruppert Structure Manifold

Because we are searching for factors of separated polynomials, we can conduct a more focused search on the Ruppert structure manifold than is possible for Gao et al. (2004). We saw in (3.2) that the (absolute) irreducibility of

$$\Phi_f = \sum_{0\le i,j\le n} \phi_{ij} x^i y^j \in \mathbb{R}[x,y], \qquad \text{for } f = \sum_{1\le i\le n} f_i x^i \in \mathbb{R}[x]$$

is indicated by the rank deficiency of

$$R_f = \sum_{0\le i,j\le n} \phi_{ij} R_{ij},$$

for some $R_{ij} \in \mathbb{Z}^{4n^2\times 2n^2+n-1}$ dependent only upon $\deg_x \Phi_f$ and $\deg_y \Phi_f$. Since

$$\Phi_f = \sum_{1\le i\le n} f_i \left(\sum_{0\le j\le i} x^j y^{i-j} \right),$$

Φ_f is absolutely irreducible if and only if

$$R_f = \sum_{1\le i\le n} f_i \left(\sum_{0\le j\le i} R_{j,i-j} \right)$$

is rank deficient. Thus, finding the nearest decomposable polynomial $\tilde{f} = \sum_{1\le i\le n} \tilde{f}_i x^i$ to f corresponds exactly to finding the nearest vector $(\tilde{f}_1,\ldots,\tilde{f}_n)$ to $(f_1,\ldots,f_n)$ such that

$$\sum_{0\le i\le n} \tilde{f}_i T_i, \quad \text{where } T_i = \sum_{0\le j\le i} R_{j,i-j} \text{ for } 1\le i\le n,$$

is rank deficient. This problem is an instance of the Structured Total Least Squares (STLS) problem, for which there are a number of established (albeit heuristic) algorithms. See, e.g., De Moor (1994), Lemmerling (1999). We examine a number of approximate polynomial problems and their formulations as STLS problems, as well as a particular heuristic (the Riemannian SVD), in Botting et al. (2005). We summarize some of the results for the approximate decomposition problem in the next section.

5. Numerical Experiments

In this section we compare our algorithm empirically with the algorithms presented in Corless, Giesbrecht, Jeffrey, and Watt (1999). They present two algorithms, both of which are numerical iterations starting from a potential right composition factor obtained by running the exact decomposition algorithm of Kozen and Landau (1989). This will not generally be a right composition factor, but they demonstrate it is often a good initial approximation. In particular, the starting point closely matches the high order coefficients of f, while ignoring the lower order coefficients until the iteration, which may well be wise if these high order coefficients are dominant, as is often the case. Their first algorithm is a fast (linear-time), linearly convergent method, while the second is a standard, and relatively expensive but quadratically convergent, Newton iteration.

In our tests we consider only decompositions of monic polynomials into monic polynomials since we want in some sense to have representative degrees for the composition factors. We want to avoid approximate decompositions in which the leading coefficients of the composition factors becomes very small when compared to the other coefficients, i.e., in which the numerical degree is not representative of the actual degree. We note that in fact the algorithm of Corless et al. (1999) will often produce such decompositions unless explicitly constrained not to.

We conduct tests on the method for estimating a radius of indecomposability, as described in Sect. 3. The lower bounds were determined for monic polynomials generated by choosing monic polynomials $g, h \in \mathbb{R}[x]$ with rational coefficients selected randomly and uniformly between -5 and 5, with 6 decimal digits (except the constant coefficient which is always 0), then composing them to form a monic $f \in \mathbb{R}[x]$. We then perturb each of the coefficients of f with uniformly distributed random noise of norm specified in the column labeled "$\|f_\Delta\|$" in the the table below (keeping the perturbed f monic with constant coefficient 0).

All results in the table are the median value of 100 runs, and are scaled by dividing through by the norm of the perturbed f. The sixth column is the median of the distance to the smallest polynomial that decomposes (computed by solving the optimization using Gröbner basis methods and the SALSA package for Maple `http://fgbrs.lip6.fr/Software/`). It is interesting to note that the smallest singular value is usually quite close to that distance. This suggests that perhaps our lower bound is too small, and that is may be possible to prove a better lower bound, perhaps $\frac{1}{\sqrt{d}}\,\sigma(\mathrm{Rup}(\Phi_f))$ (or perhaps even a constant multiple of the smallest singular value).

deg g	deg h	$\|f_\Delta\|$	$\sigma(\mathrm{Rup}(\Phi_f))$	Lwr Bnd	Abs Min
2	2	10^{-3}	2.4028e-4	5.2687e-6	1.7212e-4
2	2	10^{-1}	1.7646e-2	3.8693e-4	1.3613e-2
3	2	10^{-5}	1.0934e-6	4.8793e-9	1.1643e-6
3	2	10^{-3}	9.3678e-5	4.1802e-7	1.1935e-4
3	2	10^{-1}	7.8925e-3	3.5219e-5	1.1875e-2
2	3	10^{-5}	2.7156e-6	1.2118e-8	1.7172e-6
2	3	10^{-1}	2.7610e-2	1.2321e-4	1.9851e-2

To compare the approximate factorization algorithms, we generated random monic g and h with integer coefficients in $[-10..10]$ then composed to form $f = \sum f_i x^i$ which we perturbed with $f_\Delta = \sum \delta_i x^i$ where $\delta_i/(f_i\,\epsilon) \in [-10, 10]$ and the δ_i have 5 decimal digits. This choice for f_Δ is made to compensate for the fact that some of the coefficients of composed polynomials are significantly larger than others.

In the following table, "CGJW" indicates the iterative algorithm from Corless et al. (1999), "AppFac" indicates the approximate factorization based decomposition algorithm, and "RiSVD" indicates the Riemannian SVD based method. "Error" is the median relative error of the decomposition found over 30 or 60 runs, "Best" is the number of times this method was better than both the others (the difference between the sums of the columns and 100% were two or three way ties). In some tests, indicated by a '*' in the "Best" column of "AppFac", the best decompositions produced by the approximate factorization method could have slightly larger degree than f (the degree was not bigger in all case however). For small examples "Abs" is the number of times this method found the absolute minimum, for larger examples the value is in italics and is the number of times the result of this method was the best or tied for the best.

On average, CGJW is about 10 times faster than RiSVD which is about ten times faster than AppFac. All three algorithms find the same answer in most of the examples. Occasionally, when ϵ is relatively small, CGJW produces a very bad decomposition, while the AppFac or RiSVD still produce good results. On the other hand, when ϵ is large, the approximate factorization may not find a factor of the correct degree leading to a worse result than the other two algorithms. The approximate factorization based decomposition algorithm is also significantly slower than the other two. This suggests that in practice one should probably used the Gauss-Newton iteration algorithm from Corless et al. (1999), and revert first to RiSVD and failing that to AppFac if the backwards error of the result is not about the same order of magnitude as $\sigma(\mathrm{Rup}(\Phi_f))$.

			CGJW			AppFac			RiSVD		
deg g	deg h	ϵ	Error	Best	Abs	Error	Best	Abs	Error	Best	Abs
2	2	10^{-4}	1.53e-5	0%	100%	1.59e-5	0%	98%	1.53e-5	0%	98%
2	3	10^{-4}	1.15e-4	3%	100%	9.70e-3	0%	45%	1.26e-4	0%	97%
3	2	10^{-4}	1.90e-5	0%	96%	1.90e-5	0%	100%	1.90e-5	0%	100%
4	2	10^{-4}	1.85e-4	5%	97%	6.21e-2	2%	28%	2.20e-4	0%	88%
2	4	10^{-4}	2.99e-5	0%	98%	4.17e-5	0%	83%	2.99e-5	2%	98%
3	3	10^{-4}	1.67e-4	0%	*100%*	3.08e-4	0%	*60%*	1.67e-4	0%	*100%*
2	5	10^{-4}	4.18e-4	3%	*87%*	1.31e-1	10%	*30%*	4.18e-1	3%	*87%*
5	2	10^{-4}	3.55e-5	0%	*100%*	4.48e-05	0%	*87%*	3.55e-05	0%	*100%*
2	6	10^{-4}	1.03e-2	0%	*90%*	1.24e-1	10%*	*27%*	1.03e-2	0%	*90%*
3	4	10^{-4}	2.88e-4	0%	*100%*	1.44e-4	0%	*30%*	2.88e-4	0%	*100%*
4	3	10^{-4}	2.78e-4	0%	*100%*	8.05e-3	0%	*2%*	2.78e-4	0%	*100%*
6	2	10^{-4}	3.83e-5	0%	*100%*	7.14e-2	0%	*37%*	3.83e-5	0%	*100%*
2	2	10^{-1}	8.74e-3	2%	95%	9.18e-3	2%	83%	9.52e-3	3%	87%
2	3	10^{-1}	5.24e-2	5%	70%	5.37e-2	2%	43%	5.08e-2	12%	80%
3	2	10^{-1}	1.54e-2	0%	85%	1.47e-2	8%	95%	1.54e-2	2%	87%
2	4	10^{-1}	1.21e-1	5%	50%	1.61e-1	8%*	25%	1.21e-1	7%	53%
4	2	10^{-1}	2.50e-2	2%	43%	2.28e-2	18%*	30%	2.50e-2	3%	40%
3	3	10^{-1}	1.25e-1	0%	*40%*	6.35e-2	60%*	*97%*	1.25e-1	0%	*40%*
2	5	10^{-1}	2.04e-1	0%	*87%*	2.13e-1	13%*	*60%*	2.04e-1	0%	*87%*
5	2	10^{-1}	2.99e-2	0%	*90%*	2.99e-2	10%*	*83%*	2.99e-2	0%	*90%*
2	6	10^{-1}	2.40e-1	0%	*80%*	2.28e-1	17%*	*83%*	2.40e-1	3%	*83%*
3	4	10^{-1}	1.87e-1	0%	*63%*	1.34e-1	37%*	*93%*	1.87e-1	0%	*63%*
4	3	10^{-1}	8.20e-2	0%	*83%*	7.69e-2	10%*	*80%*	7.54e-2	7%	*90%*
6	2	10^{-1}	2.42e-2	0%	*93%*	4.63e-2	7%	*63%*	2.42e-2	0%	*93%*
2	2	1	2.40e-2	0%	95%	2.40e-2	0%	93%	2.40e-2	0%	95%
2	3	1	1.83e-1	2%	75%	1.74e-1	7%*	70%	1.79e-1	7%	75%
3	2	1	1.62e-1	0%	77%	1.51e-1	7%*	77%	1.44e-1	8%	87%
2	4	1	3.78e-1	2%	50%	3.45e-1	30%*	28%	3.70e-1	8%	55%
4	2	1	2.87e-1	8%	58%	2.63e-1	22%*	40%	2.89e-1	5%	60%
3	3	1	4.98e-1	0%	*40%*	3.08e-1	43%*	*93%*	4.69e-1	7%	*57%*
2	5	1	4.02e-1	7%	*53%*	3.58e-1	33%*	*70%*	4.09e-1	13%	*57%*
5	2	1	3.80e-1	10%	*60%*	3.43e-1	20%*	*80%*	3.80e-1	0%	*70%*
2	6	1	4.53e-1	7%	*60%*	4.44e-1	23%*	*73%*	4.49e-1	17%	*70%*
3	4	1	4.54e-1	17%	*57%*	4.34e-1	27%*	*53%*	4.43e-1	17%	*57%*
4	3	1	4.13e-1	3%	*60%*	3.54e-1	33%*	*83%*	3.98e-1	7%	*63%*
6	2	1	4.09e-1	0%	*77%*	3.83e-1	17%*	*77%*	4.09e-1	7%	*83%*

Acknowledgment

We would like to thank Erich Kaltofen for early important discussions related to this work, and for later comments. The first author would like to thank the NSERC and MITACS. Some of this work was done while the authors were visiting the Key Laboratory of Mathematics-Mechanization, Institute of Systems Science, Chinese Academy of Sciences in Beijing. We thank them for their generous support.

References

C. Alonso, J. Gutiérrez and T. Recio. A rational function decomposition algorithm by near-separated polynomials. *J. Symb. Comput.*, 19(6): 527–544, 1995.

D.R. Barton and R. Zippel. A polynomial decomposition algorithm. In *SYMSAC '76: Proc. Third ACM Symposium on Symbolic and Algebraic Computation*, pages 356–358, New York, NY, USA, 1976. ACM Press.

D.R. Barton and R. Zippel. Polynomial decomposition algorithms. *J. Symb. Comput.*, 1: 159–168, 1985.

B. Botting, M. Giesbrecht and J.P. May. Using the Riemannian SVD for problems in approximate algebra. In *SNC 2005: Proc. Internat. Workshop on Symbolic-Numeric Computation*, pages 209–219, Xi'an, China, 2005.

R.M. Corless, M. Giesbrecht, D. Jeffrey and S.M. Watt. Approximate polynomial decomposition. In *ISSAC '99: Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.*, pages 213–220, 1999. ACM Press.

B. De Moor. Total least squares for affinely structured matrices and the noisy realization problem. *IEEE Transactions on Signal Processing*, 42: 3004–3113, 1994.

M.D. Fried. On a conjecture of Schur. *Mich. Math. Journal*, 17: 41–55, 1970.

M.D. Fried and R.E. MacRae. On the invariance of chains of fields. *Ill. J. Math.*, 13: 165–171, 1969.

S. Gao, E. Kaltofen, J. May, Z. Yang and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *ISSAC 2004: Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.*, pages 167–174, 2004. ACM Press.

J. von zur Gathen. Functional decomposition of polynomials: the tame case. *J. Symb. Comput.*, 9:281–299, 1990.

J. Gutiérrez, T. Recio and C. Ruiz de Velasco. Polynomial decomposition algorithm of almost quadratic complexity. In *Proc. Applied Algebra, Algebraic Algorithms and Error-correcting Codes*, volume 357 of *Lecture Notes in Comp. Sci.*, pages 471–475, Rome, 1988. Springer, Berlin.

E. Kaltofen and J. May. On approximate irreducibility of polynomials in several variables. In *ISSAC 2003: Proc. 2003 Internat. Symp. Symbolic Algebraic Comput.*, pages 161–168, 2003. ACM Press.

D. Kozen and S. Landau. Polynomial decomposition algorithms. *J. Symb. Comput.*, 7: 445–456, 1989.

P. Lemmerling. *Structured total least squares: analysis, algorithms and applications.* PhD thesis, K.U. Leuven (Leuven, Belgium), 1999.

J. Renegar. On the computational complexit and geometry of the first-order theory of the reals: parts I, II, III. *J. Symb. Comput.*, 13(3): 255–352, 1992.

W.M. Ruppert. Reducibility of polynomials *f(x,y)* modulo *p*. *J. Number Theory*, 77: 62–70, 1999.

G. Turnwald. On Schur's conjecture. *J. Austral. Math. Soc. Ser. A*, 58(3): 312–357, 1995.

R. Zippel. Rational function decomposition. In *Proc. ISSAC '91: Proc. 1991 Internat. Symp. Symbolic Algebraic Comput.*, pages 1–6, 1991. ACM Press.

Mark Giesbrecht and John May
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
e-mail: `mwg@uwaterloo.ca`
`jpmay@uwaterloo.ca`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 113–129
© 2007 Birkhäuser Verlag Basel/Switzerland

Amortized Bound for Root Isolation via Sturm Sequences

Zilin Du, Vikram Sharma and Chee K. Yap

Abstract. This paper presents two results on the complexity of root isolation via Sturm sequences. Both results exploit amortization arguments.

For a square-free polynomial $A(X)$ of degree d with L-bit integer coefficients, we use an amortization argument to show that all the roots, real or complex, can be isolated using at most $O(dL + d \lg d)$ Sturm probes. This extends Davenport's result for the case of isolating all real roots.

We also show that a relatively straightforward algorithm, based on the classical subresultant PQS, allows us to evaluate the Sturm sequence of $A(X)$ at rational $\widetilde{O}(dL)$-bit values in time $\widetilde{O}(d^3L)$; here the $\widetilde{O}$-notation means we ignore logarithmic factors. Again, an amortization argument is used. We provide a family of examples to show that such amortization is necessary.

Mathematics Subject Classification (2000). Primary 68Q25; Secondary 34A34; Tertiary 26C10.

Keywords. Sturm sequence, Davenport-Mahler bound, subresultant, complexity, root isolation, separation bound.

1. Introduction

Sturm sequences are a classical tool for real root isolation [CL83]. We recall the main steps of the standard real root isolation algorithm based upon Sturm sequences [CL83]: Let $A(X)$ be a square-free integer polynomial of degree d with L-bit coefficients.

(1) Compute the Sturm sequence of $A(X)$.
(2) Compute an interval $(-B, B)$ containing all real zeros of $A(X)$. Initialize a queue Q with the interval $(-B, B)$.

The work is supported by NSF Grant #CCF-043836. A preliminary version of this work appeared at the International Workshop on Symbolic-Numeric Computation (SNC 2005), Xi'an, China, July 19–21, 2005.

(3) While Q is non-empty, do: extract an interval I from Q and compute the number of zeros in I using the Sturm sequence. If I has one zero, output I. If I has no zeros, discard I. Otherwise, split I into two open intervals I_L, I_R, at the midpoint $m(I)$ of I. Check if $m(I)$ is a zero. If so, output $m(I)$. Push I_L and I_R into Q.

The original bound for this algorithm is $\widetilde{O}(d^7L^3)$ from Collins-Loos [CL83]. Davenport [Dav85, Prop. 3'] stated a complexity bound of $\widetilde{O}(d^4L^2)$. In this paper, the $\widetilde{O}$-notation means that we are ignoring logarithmic factors. These complexity bounds are estimated from the following three bounds:

(I) The complexity of computing the Sturm sequence.
(II) The complexity of evaluating a Sturm sequence at a given point.
(III) Bound on the number of bisections needed to isolate all the roots .

The complexity of (I) is $\widetilde{O}(d^4L^2)$ in [CL83], but this has been improved to $\widetilde{O}(d^2L)$ in [LR01, Rei97]. The best complexity bound for (II) is $\widetilde{O}(d^3L)$ assuming (as we may in root isolation) that the evaluation point is a rational number with bit size $\widetilde{O}(dL)$. Although Davenport [Dav85] stated this bound, the first published proofs was given by Reischert [Rei97], and independently, by Lickteig-Roy [LR01]. The bound for (III) is $\widetilde{O}(d^2L)$ in Collins-Loos [CL83]. Davenport improved the bound in (III) by a factor of d to $O(dL + d\lg d)$. The overall complexity of real root isolation using Sturm sequences is the product of the bounds in (II) and (III). Thus the best current bound for real root isolation via Sturm sequences is $\widetilde{O}(d^4L^2)$.

Our Contribution. In this paper we give amortization arguments which achieve the above bounds for (II) and (III). Both results use amortization analysis, a technique that is common in discrete algorithms [CLRS01, Chap. 17].

For (II), we give an approach that is much simpler than Reischert or Lickteig-Roy. We rely only on the standard theory of Subresultant polynomial remainder sequences (PRS). But instead of the PRS, we use another idea that goes back to Strassen [Str83], which represents the PRS by its polynomial quotient sequence (PQS). We then give an amortized argument for the straightforward evaluation of this PQS. We also give a family of examples to show that a non-amortized worst case bound will not do.

For (III), we give a charging scheme argument that leads to a slightly sharper bound than that of Davenport. But the main benefit of our argument is its extendibility to the case of isolating complex roots of a polynomial; it is not obvious how to extend Davenport's argument to this case. In particular, we show that the number of Sturm sequence evaluations are $O(dL + d\lg d)$ even for the case of isolating complex roots of $A(X)$. We think our argument gives some insights into how the distances between various roots actually affect the complexity of Step (III). There is a key difference between Sturm's method applied to isolating complex roots, for instance in [Pin76, Wil78], as opposed to the case of real roots: in the complex case, one has to re-compute a Sturm sequence at each probe. This drawback can be overcome by applying the two-dimensional Sturm sequences of Hermite (cf. Pedersen [Ped90], or an alternative by Milne [Mil92]).

The Complexity of Root Isolation. The complexity results for root isolation via Sturm sequences is inferior to the $\widetilde{O}(d^3L)$ bound obtained by Schönhage [Sch82]. Nevertheless, there are some advantages in the Sturm approach: Schönhage's algorithm simultaneously approximates all the roots of a polynomial, but these approximations may not represent isolations until the root separation bound is achieved. In contrast, the Sturm approach can isolate any subset of roots in a suitable region, or the i'th largest real root for any chosen i or range of i's. The Sturm method is ideally suited for root isolation, a problem that is distinct from root approximation.

There are many other results [Ren87, KS94, NR96, Pan96] on the complexity of root approximation that do not directly depend on root isolation. These methods, like Schönhage's, simultaneously approximate all the complex roots of a polynomial. For instance, the bit complexity for the Neff-Pan algorithm is $\widetilde{O}(d^3L+d\mu)$ where μ is the desired relative precision in each complex zero. If we choose $\mu = \widetilde{O}(dL)$ which is the root separation bound, we are assured of isolating all the roots. In any case, these bounds do not improve Schönhage's bound. In comparing complexity bounds, we must take in account normalization assumptions. E.g., Pan [Pan96] normalized the polynomials so that all its zeros lie in the unit circle. This transforms a polynomial $A(X)$ with L-bit coefficients into a normalized polynomial with dL-bit coefficients.

2. Efficient Evaluation of Sturm Sequences: Simplified Approach

In this section, we address the complexity of Step (II) in the introduction. In particular, we must evaluate Sturm sequences at rational values of X with bit sizes proportional to the logarithm of the root separation bound; the latter we know is bounded by $O(d(L+\lg d))$. Also, a rational number has L-bits if its numerator and denominators are at most L-bit integers.

Recall that Reischert [Rei97] and Lickteig-Roy [LR01] have showed the complexity of Step (II) as $\widetilde{O}(d^3L)$. However, their approaches are fairly complicated and require specialized algorithms (Reischert uses a generalized form of the half-GCD algorithm and Lickteig-Roy use computation over the rational field $\mathbb{Q}$). We now show how a fairly straightforward algorithm that achieves the same bounds.

Let $A(X), B(X) \in \mathbb{Z}[X]$ where $d = \deg(A) > \deg(B)$ and the bit lengths of the coefficients of A, B are at most L. Recall the notion [Yap00, p. 83] of a **polynomial remainder sequence** (PRS) of (A,B) **based on a sequence** $(\beta_1,\ldots,\beta_{k-1})$ where $\beta_i \in \mathbb{Z}$: this is a sequence

$$(A_0, A_1, \ldots, A_k) \tag{2.1}$$

of polynomials such that $A_0 = A$ and $A_1 = B$, and for $i = 1, \ldots, k$, there exists $Q_i \in \mathbb{Z}[X]$ such that

$$\beta_i A_{i+1} = a_i^{\delta_i+1} A_{i-1} - Q_i A_i \tag{2.2}$$

where a_i is the leading coefficient of A_i and $\delta_i = \deg(A_{i-1}) - \deg(A_i) = \deg(Q_i) > 0$, with the termination condition that $A_{k+1} = \beta_k = 0$. The key problem in PRS is to devise effective methods for computing the β_i's so that the bit size of coefficients of the A_i remain polynomial in d and L. In particular, the **subresultant PRS** from Collins [Yap00, p. 89] achieves this with bit sizes of coefficients bounded by $\widetilde{O}(dL)$.

Let the "bit size" of $A(X)$ be $\lg H(A(X))$ where $H(A(X))$ is the **height** of $A(X)$, i.e., the maximum of the absolute value of the coefficients of $A(X)$ [Yap00, p. 23]. An alternative representation for the PRS uses the following concept. Let us define the **polynomial quotient sequence** (PQS) of (A, B) **based on a sequence** $(\beta_1, \ldots, \beta_{k-1})$ to be a sequence

$$(A_0, A_1, Q_1, Q_2, \ldots, Q_{k-1}) \tag{2.3}$$

where the Q_i's are defined as in (2.2). Note that the number of coefficients in the PRS (2.1) may be $\Omega(d^2)$. Thus if it is used as a Sturm sequence for A_0, A_1, evaluating this sequence at any value of X may require $\Omega(d^2)$ arithmetic operations. In contrast, if we only store the PQS in (2.3) and also $(\beta_1, \ldots, \beta_{k-1})$, we can easily evaluate the Sturm sequence at any X using only $O(d)$ arithmetic operations; the reason is that $\sum_{i=1}^{k-1} \deg(Q_i)$ is only d. This advantage in the number of arithmetic operations has been noted by many authors including [LR01, Rei97]. However, when we consider bit complexity, it is no longer clear whether we still have an advantage by a factor of d.

It is not hard to see from the definition of a PQS that the bit sizes of the Q_i's are $\widetilde{O}(d^2 L)$. More precisely:

Lemma 1. *The bit sizes of the coefficients $Q_i(X)$ is bounded by $O(\delta_i dL)$.*

Proof. From (2.2) we know that Q_i is the pseudo-quotient of A_{i-1} divided by A_i. By [Yap00, Lemma 3.8], the coefficients of Q_i is obtained as principal minors of a matrix M of size $\delta_i + 1 \times \deg(A_{i-1}) + 1$. The first row of M contains the coefficients of A_{i-1} and the remaining rows contains shifted coefficients of A_i. Since each entry has $\widetilde{O}(dL)$ bits, the minors have the stated bound. $\square$

The following example shows that the bit sizes of coefficients of the Q_i's can be $\Omega(d^2 L)$. Consider the subresultant PRS for the polynomials

$$A_0(X) = aX^{3d}, \qquad A_1(X) = bX^{2d} + c.$$

Then we have $A_2(X) = (-1)^d b^d ac X^d$, $\beta_1 = (-1)^{d+1}$, $Q_1(X) = b^d a X^d$, $A_3(X) = (-1)^d b^{d-1} (ac)^{d+1}$, $\beta_2 = (-1)^d b^{d^2+1}$. $Q_2 = b^{d^2+1} (ac)^d X^d$.

It follows that a naive worst case bound of $\widetilde{O}(dL)$ on the coefficients of the PQS is wrong.

Now we show the desired bound on the complexity of evaluating Sturm sequences using PQS.

Theorem 2. *Let $A, B \in \mathbb{Q}[X]$ have degree d, and let its coefficients be L-bit rationals. Let $(A_0, A_1, \ldots, A_h)$ be the subresultant PRS of A, B, based on $(\beta_1, \ldots, \beta_{h-1})$. Also, let a_i be the leading coefficient of A_i.*

(i) *We can compute $(A_0, \ldots, A_h)$, $(\beta_1, \ldots, \beta_{h-1})$ and also $(a_0, \ldots, a_h)$ in time $\widetilde{O}(d^3L)$.*
(ii) *We can evaluate the Subresultant PRS of A, B at any $\widetilde{O}(dL)$-bit rational value in time $\widetilde{O}(d^3L)$.*

Proof.

(i) This is done by a straightforward evaluation of the subresultant PRS [Yap00, Sect. 3.5, p. 89].
(ii) Let x be an $\widetilde{O}(dL)$-bit rational. To compute $A_i(x)$ for $i = 0, \ldots, h$, we use the following steps:
Step (a): Evaluate $A_0(x), A_1(x)$.
Step (b): Evaluate $Q_1(x), \ldots, Q_{h-1}(x)$.
Step (c): For $i=1$ to $h-1$, compute $A_{i+1}(x)$ as $(a_i^{\delta_i+1} A_{i-1}(x) - Q_i(x)A_i(x))/\beta_i$.

All the polynomial evaluations must be done using Horner's rule in order for our bounds to be valid. Step (a) is $\widetilde{O}(d^2L)$. For Step (b), the complexity of evaluating Q_i at x is $\widetilde{O}(\delta_i^2 dL)$ (by Lemma 1). Summing over all i's, we obtain an overall complexity of $\sum_{i=1}^{h-1} \widetilde{O}(\delta_i^2 dL) = \widetilde{O}(d^3L)$, since $\sum_{i=1}^{h-1} \delta_i \le d$. Finally, for step (c), we note that each of the quantities $a_i^{\delta_i+1}$, $Q_i(x)$, $A_i(x)$, and hence β_i, is $\widetilde{O}(d^2L)$-bit rationals. Thus, the cost of computing each $A_i(x)$ is $\widetilde{O}(d^2L)$ and so the overall cost of step (c) is $\widetilde{O}(d^3L)$. □

The above proof amounts to a simple algorithm for achieving $\widetilde{O}(d^3L)$ bit complexity for Step (II). The amortization argument amounts to exploiting the inequality $\sum_i \delta_i \le d$.

3. The Davenport-Mahler Bound

The basic inequality that our amortized analysis will exploit is the Davenport-Mahler theorem (Theorem 3). This theorem gives a lower bound on the product of differences of certain pairs of roots of a polynomial $A(X) = a_d \prod_{i=1}^{d}(X - \alpha_i)$ in terms of its **discriminant** $\operatorname{disc}(A) = a_d^{2d-2} \prod_{1 \le i < j \le n}(\alpha_i - \alpha_j)^2$ and **Mahler measure** $\mathrm{M}(A) = |a_d| \prod_{i=1}^{d} \max\{1, |\alpha_i|\}$, see [Yap00, 6.6, 4.5] [Mc99, 1.5, 2.1]. The literature has several variants of this theorem that use the same proof but formulate different conditions on how roots may be paired so that the proof works. We give the most general condition supported by the proof. It is equivalent to Johnson's formulation [Joh98] and generalizes Davenport's original formulation [Dav85, Prop. I.5.8].

Theorem 3. *Let $A(X) = a_d \prod_{i=1}^{d}(X - \alpha_i)$ be a square-free complex polynomial of degree d. Let $G = (V, E)$ be a directed graph whose nodes $\{v_1, \ldots, v_k\}$ are a subset of the roots of $A(X)$ such that:*

1. *If $(\alpha, \beta) \in E$ then $|\alpha| \le |\beta|$;*
2. *G is acyclic;*
3. *The in-degree of any node is at most 1.*

If exactly m of the nodes have in-degree 1, then

$$\prod_{(v_i,v_j)\in E} |v_i - v_j| \ge \sqrt{|\mathrm{disc}(A)|} \cdot \mathrm{M}(A)^{-(d-1)} \cdot (d/\sqrt{3})^{-m} \cdot d^{-d/2}. \tag{3.1}$$

Proof. See [ESY06, Theorem 3.1]. □

Remark 3.1. Suppose the edge set of a graph $G = (V, E)$, $V \subseteq \{\alpha_1, \dots, \alpha_d\}$, can be partitioned into k disjoint edge sets $E = E_1 \cup \cdots \cup E_k$ such that the each of the graphs $G_1 = (V, E_1), \dots, G_k = (V, E_k)$ satisfies the properties in the theorem above, then $\prod_{(u,v)\in E} |u - v|$ is bounded from below by the product of the bounds corresponding to each G_i.

The following lemma gives us an upper and lower bound on the product of k intervals defined by the real roots of a polynomial $A(X)$. For any polynomial $A(X)$, let $\mathrm{lead}(A)$ be its leading coefficient.

Lemma 4. *Let $A(X) \in \mathbb{R}[X]$ be a square-free polynomial of degree d. If $\alpha_1 < \beta_1 \le \alpha_2 < \beta_2 \le \alpha_3 < \cdots < \beta_k$ are real zeros of $A(X)$ then*

$$\prod_{i=1}^{k} |\alpha_i - \beta_i| \begin{cases} \le & \mathrm{M}(A)/\mathrm{lead}(A), \\ \ge & \mathrm{M}(A)^{-d+1} d^{-d/2} (\sqrt{3}/d)^k. \end{cases} \tag{3.2}$$

Proof. Let $\gamma_1, \dots, \gamma_d$ be all the (not necessarily distinct) zeros of $A(X)$. First we prove that $\mathrm{M}(A)$ is an upper bound on the product $\prod_{i=1}^{k} |\alpha_i - \beta_i|$. We consider two possibilities.
Case A: Suppose there exists an $h = 1, \dots, k$ such that $\alpha_h < 0 < \beta_h$. Let $\gamma_1, \dots, \gamma_d$ denote all the distinct roots in the set $\{\alpha_i, \beta_i : i = 1, \dots, k\}$. Thus, $k+1 \le d \le 2k$. We have

$$\begin{aligned} \prod_{i=1}^{k} |\beta_i - \alpha_i| &= \left(\prod_{i=1}^{h-1} (\beta_i - \alpha_i)\right) \cdot |\alpha_h - \beta_h| \cdot \left(\prod_{i=h+1}^{k} (\beta_i - \alpha_i)\right) \\ &\le \left(\prod_{i=1}^{h-1} |\alpha_i|\right) \cdot (|\alpha_h| + |\beta_h|) \cdot \left(\prod_{i=h+1}^{k} |\beta_i|\right) \\ &\le (\prod_{i=1}^{d} \max\{1, |\gamma_i|\}|) \end{aligned}$$

which, in turn, is bounded by $\mathrm{M}(A)/|\mathrm{lead}(A)|$.
Case B: Suppose $\beta_i \le 0 \le \alpha_{i+1}$ for some $i = 0, 1, \dots, k+1$ (with $\beta_0 = -\infty, \alpha_{k+1} = \infty$). The above argument can easily be adapted to this case as well.

The lower bound follows from [Dav85, Prop. 8]. □

4. Amortized Bound on Number of Probes to Isolate Real Roots

Suppose (a, b) is an open interval containing k **roots**,

$$a < \alpha_1 < \alpha_2 < \cdots < \alpha_k < b. \tag{4.1}$$

The values of these roots are unknown, and our goal is to "locate" them by isolating intervals. We will bound the size of the binary search tree of the algorithm described in introduction in terms of the amortized bound given in (3.2). The bounds expressed in (3.1) and (3.2) are amortized because they are better than the worst case bound obtained by taking the product of the worst case for each gap, i.e., the root separation bound. We next formalize our problem and give a general framework of an algorithm that encompasses the one based upon Sturm sequences.

All our intervals are either open intervals $I = (c, d)$ or exact intervals $I = [c, c]$ for some $a \leq c < d \leq b$. The **width** of I is $w(I) = d - c$ for open intervals and $w(I) = 0$ for exact intervals. Also, let $\#(I)$ denote the number of roots in I. If $I = [c, c]$, we write $\#(c)$ instead of $\#([c, c])$. Clearly $\#(c) = 0$ or 1. We call I an **isolating interval** if $\#(I) = 1$.

The **Real Root Isolation Problem** for an interval $I = (a, b)$ is that of finding a set of $\#(I)$ pairwise disjoint isolating intervals containing the real roots in I. To solve this problem, we consider algorithm that make "probes". Each **probe** is defined by an input open interval I and the **result** of the probe is the pair $(\#(I_L), \#(I_R))$ where $m = m(I) = (c + d)/2$, $I_L = (c, m)$ and $I_R = (m, d)$. Note that $\#(m) = \#(I) - \#(I_L) - \#(I_R)$.

In the following, let $\tau > 0$ be an arbitrary **threshold parameter**. For instance, we may choose $\tau = 1$. Relative to τ, we define an interval I as **small** or **big** depending on whether $w(I) < \tau$ or $w(I) \geq \tau$.

Let I have roots as in (4.1). A **segment** of I is an interval of the form $\sigma_i = (\alpha_i, \alpha_{i+1})$ for $i = 0, 1, \ldots, k$, with $a = \alpha_0$ and $b = \alpha_{k+1}$. Let $\overline{\sigma}(I) = \{\sigma_0, \sigma_1, \ldots, \sigma_k\}$ denote the set of segments of I. Call σ_0 and σ_k the **outer segments**; all others are **inner segments**. Define $\Delta(I) = \Delta^+ + \Delta^-$ where

$$\begin{aligned} \Delta^+ = \Delta^+(I) &:= \sum_{\sigma_i \text{ is big}} 2\lg(w(\sigma_i)/\tau), \\ \Delta^- = \Delta^-(I) &:= \sum_{\substack{\sigma_i \text{ is a small} \\ \text{inner segment}}} (1 - \lg(w(\sigma_i)/\tau). \end{aligned} \tag{4.2}$$

Thus, Δ^+, Δ^- are the summations over big and small segments (respectively). By definition, Δ^+ (resp., Δ^-) is equal to 0 if there are no big (resp., small) segments.

Theorem 5. *Given an interval $I = (a, b)$ and also $k = \#(I)$, we can solve the real root isolation problem for I using at most $k - 1 + \Delta(I)$ probes.*

Proof. We first present the algorithm. This is a standard binary search: if $\#(I) \leq 1$, the problem is trivial. Otherwise, we initialize a queue to contain just the interval

$I = (a, b)$. In the general step, we extract an interval $J = (c, d)$ from the queue and probe J. The probe returns the pair $(\#(J_L), \#(J_R))$ as above. Inductively, we may assume that $\#(J)$ is known. If $\#(J_L) + \#(J_R) < \#(J)$, we can output $m(J)$ as a root. For each $i = L, R$, we have three possibilities: if $\#(J_i) = 0$, we discard J_i; if $\#(J_i) = 1$, we output J_i; if $\#(J_i) > 1$, we put J_i into the queue. The algorithm halts when the queue is empty. It is clear that the output is a complete list of isolating intervals.

We now prove that this algorithm halts after at most $k - 1 + \Delta(I)$ probes using an amortization argument. Let J be an interval that was probed by the algorithm. Probe J produces the subintervals J_L, J_R. We call J a **splitting** probe if $m(J)$ is a root, or if both J_L and J_R are placed in the queue; otherwise J is **non-splitting**. Clearly, there are at most $k-1$ splitting probes. It remains to prove that at most $\Delta(I)$ probes are non-splitting. This is done by introducing a charging scheme for such probes.

Let $J = (c, d)$ be a non-splitting probe. Then $m(J) = (c+d)/2$ belongs to an outer segment $\sigma \in \overline{\sigma}(J)$. We shall charge probe J to a segment $\sigma' \in \overline{\sigma}(I)$, defined as follows:

$$\sigma' = \begin{cases} \text{the segment in } \overline{\sigma}(I) \text{ that contains } \sigma & \text{if } w(\sigma) \geq \tau, \\ \text{the inner segment in } \overline{\sigma}(J) \text{ adjacent to } \sigma & \text{if } w(\sigma) < \tau. \end{cases}$$

σ' has the following properties:

- It is uniquely defined.
- It is a segment in $\overline{\sigma}(I)$, even when $w(\sigma) < \tau$.
- It is big if σ is big, which is clear.
- It is small if σ is small: to see this, note that $w(\sigma') \subset w(J)$ and also $w(J) = 2w(\sigma) < 2\tau$. Note that σ' is an inner segment.

We now consider two cases:

Case (A): σ' is small. We show that σ' is charged at most $1 - \lg(w(\sigma')/\tau)$ times. Let $J_1, \ldots, J_\ell$ be the probe (intervals) that are charged to σ', and we may assume $\sigma' \subset J_1 \subset J_2 \subset \cdots \subset J_\ell$. Thus

$$w(\sigma') < w(J_1) \leq 2^{-1}w(J_2) \leq \cdots \leq 2^{-\ell+1}w(J_\ell) < 2^{-\ell+1}\tau.$$

Hence $\ell < 1 - \lg(w(\sigma')/\tau)$.

Case (B): σ' is big. We show that σ' is charged at most $2\lg(w(\sigma')/\tau)$ times. At the first probe J whereby $m(J) \in \sigma'$, the segment σ' is split into two halves σ'_L and σ'_R. By symmetry, consider σ'_L. Subsequently, suppose $J_1, \ldots, J_\ell$ are all the probe (intervals) that are charged to σ' via σ'_L. More precisely, assume $\sigma'_L \subset J_1$ and $J_\ell \subset J_{\ell-1} \subset \cdots \subset J_1$. Then we have

$$w(\sigma'_L) = w(J_1 \cap \sigma') \geq 2w(J_2 \cap \sigma') \geq \cdots \geq 2^{\ell-1}w(J_\ell \cap \sigma') \geq 2^{\ell-1}\tau.$$

This proves that $\ell \leq \lg(2w(\sigma'_L)/\tau)$. By also taking into account the charges via σ'_R, the total charges on σ' is at most $\lg(4w(\sigma'_L)w(\sigma'_R)/\tau^2) \leq 2\lg(w(\sigma')/\tau)$, using the fact that $w(\sigma'_L) + w(\sigma'_R) = w(\sigma')$. $\square$

Implementing the probe model. We discuss how our probe model can be implemented. If $J = (c, d)$ and $\#(J)$ is known, then the probe amounts to performing a "Sturm query" for the interval $[c, m(J)]$ and returns $(\#(J_L), \#(J_R))$. This means that we compute the Sturm sequence for the polynomial $A(X)$ and evaluate the number of sign variations at c and at $m(J)$, and take their difference. Let $V(c)$ denote the number of sign variations of the Sturm sequence evaluated at c. Indeed, if we assume that inductively, we already know $V(c)$ and $V(d)$, then this probe can be done by just computing $V(m(J))$.

Normally, the sign variation $V(x)$ is assumed to be well-defined only when x is not a root of $A(X)$. However, in case $A(X)$ is square-free, $V(x)$ is well-defined even when x is a root. To see this, let $(A_0(X), A_1(X), \ldots, A_h(X))$ be the Sturm sequence with $A_0(X) = A(X)$, $A_1(X) = dA/dX$. Then $V(x)$ is the number of sign variations in the sequence of numbers $(A_0(x), A_1(x), \ldots, A_h(x))$. But even when $A_0(x) = 0$, the sequence cannot vanish identically because $A(X)$ is square-free. Moreover, since $\operatorname{sign}(A_0(x^+)) = \operatorname{sign}(A_1(x))$, we have

$$\operatorname{sign}(A_0(x^+), A_1(x^+), \ldots, A_h(x^+)) = \operatorname{sign}(A_1(x), A_1(x), \ldots, A_h(x))$$

and hence $V(x^+) = V(x)$. Similarly, $V(x^-) = 1 + V(x^+)$. Hence $\#(J) = V(c^+) - V(d^-)$ where $J = (c, d)$. Thus our probe model can be implemented with a single sign-variation computation.

5. Complexity of Real Root Isolation

Using the bounds from the previous section we derive an a priori bound on the number of Sturm probes as given in Theorem 5.

Theorem 6. *Let $A(X) \in \mathbb{R}[X]$ be a square-free polynomial of degree d. Then we can isolate all the real roots of $A(X)$ using at most*

$$1.5d \lg d + (d+1) \lg \mathrm{M}(A) + 2d + \frac{1}{2} \lg \frac{1}{|\operatorname{disc}(A)|} + \lg \frac{\mathrm{M}(A)}{|\operatorname{lead}(A)|}$$

probes.

Proof. Without loss of generality, we can assume that $w(I)$, where $I = (-B, B)$ is the initial interval, is bounded by $2\,\mathrm{M}(A)/|\operatorname{lead}(A)|$. In this proof, we further assume $\tau = 1$. Let $\alpha_1 < \cdots < \alpha_k$ be the k real zeros of $A(X)$. The segments $\overline{\sigma}(I)$ are defined by these α's and also the two outer segments $(-B, \alpha_1)$ and (α_k, B). If any of these outer segments has width $< \tau$, then they are never charged. It is also not hard to see that the total combined charges to these 2 outer segments is at most $\lg(\mathrm{M}(A)/|\operatorname{lead}(A)|) + 1$, and they are counted as part of Δ^+.

We first invoke the upper bound in Lemma 4: choose the intervals (α_i, β_i) (for $i = 1, \ldots, m$) in this Lemma to correspond to the big segments in $\overline{\sigma}(I)$, not counting any big outer segments; the upper bound in Lemma 4 then implies

$$\prod_{i=1}^{m} |\alpha_i - \beta_i| \leq \mathrm{M}(A)/\operatorname{lead}(A).$$

But the definition of Δ^+ includes any big outer segments and from the preceding remarks we know that the outer segments are charged at most $1 + \lg \frac{\mathrm{M}(A)}{|\operatorname{lead}(A)|}$. Hence we conclude that $\Delta^+ \leq (\lg \frac{\mathrm{M}(A)}{|\operatorname{lead}(A)|} + 1) + 2\lg \mathrm{M}(A)$.

We next invoke the lower bound in Lemma 4 by choosing the intervals (α_i, β_i) (for $i = 1, \ldots, m$), $m \leq k$, in this Lemma to correspond to the small segments in $\overline{\sigma}(I)$, not counting any small outer segments. Then the lower bound in Lemma 4 implies $\Delta^- \leq (k-1) + (d-1)\lg \mathrm{M}(A) + (3d/2)\lg d + \frac{1}{2}\lg \frac{1}{|\operatorname{disc}(A)|}$. Therefore,

$$\begin{aligned}(k-1) + \Delta &\leq (d-1) + \Delta^+ + \Delta^- \\ &\leq (d-1) + \lg \frac{\mathrm{M}(A)}{|\operatorname{lead}(A)|} + 2\lg \mathrm{M}(A) + 1 + \\ &\quad (d-1) + (d-1)\lg \mathrm{M}(A) + 1.5d\lg d + \frac{1}{2}\lg \frac{1}{|\operatorname{disc}(A)|} \\ &< (d+1)\lg \mathrm{M}(A) + 1.5d\lg d + 2d + \lg \frac{\mathrm{M}(A)}{|\operatorname{lead}(A)|} + \frac{1}{2}\lg \frac{1}{|\operatorname{disc}(A)|}.\end{aligned}$$

□

Corollary 7. *We can isolate all the real roots of a square-free integer polynomial of degree d and coefficients of bit length L using at most $dL + 2d\lg d + O(d+L)$ Sturm probes.*

Proof. We have the following observations for $A(X)$:

1. From Landau's inequality (cf. [Yap00, Lem. 4.14(i)]) $\mathrm{M}(A) \leq \|A\|_2$; furthermore $\|A\|_2 \leq (d+1)^{\frac{1}{2}} 2^L$.
2. Since $A(X)$ is an integer polynomial $|\operatorname{lead}(A)| \geq 1$ and as it is square-free $|\operatorname{disc}(A)| \geq 1$.

Applying these observations to the result of the above theorem gives the following upper bound on the number of Sturm probes:

$$1.5d\lg d + (d+1)[\frac{1}{2}\lg(d+1) + L] + 2d + \frac{1}{2}(d+1) + L.$$

But this is clearly bounded by $dL + 2d\lg d + O(d+L)$. □

Remark 5.1. This result saves a term of dL as compared to [Dav85, Prop. 2].

From Theorem 2 we have the complexity of evaluating the Sturm sequence of $A(X)$ at an input of bit size dL as $\widetilde{O}(d^3 L)$.

Hence we have following bit complexity for real root isolation:

Corollary 8. *If $A(X)$ is a degree d square-free integer polynomial with L-bit coefficients, then we can isolate all the real roots of $A(X)$ in time $\widetilde{O}(d^4 L^2)$.*

Remark 5.2. Is our choice of $\tau = 1$ optimal? Consider the two extreme choices. If we choose τ to be 2^{L+1}, then all segments are small. Thus,

$$\Delta = \Delta^- \leq (d-1)(2 + 2L + 0.5\lg d) + (3d/2)\lg d.$$

This is asymptotically the same as the bound in our theorem, but slightly worse with an additive term of dL. If we choose τ to be the root separation bound, i.e., $\tau = 2^{-Ld}d^{-d}$, then all segments are big. Hence we have

$$\Delta = \Delta^{+} \leq 2L + \lg d + 2Ld^2 + 2d^2 \lg d.$$

This has an additive $d^2(L + \lg d)$ term, which is asymptotically worse than our theorem. At any rate, this suggests that τ should be chosen to have a balance between the number of big and small segments.

6. Amortized Bound on Number of Probes to Isolate Complex Roots

We can extend the result of Sect. 4 to the case of isolating any set of roots in a rectangular region S in the complex plane. More precisely, S is a half-open set including its northern and western edges but omitting its southern and eastern edges, i.e., the set $\{(x, y) : a \leq x < b, c < y \leq d\}$.

Let $w(S)$ represent the edge length of S and $\#(S)$ the number of roots in S. We call S an **isolating square** if $\#(S) = 1$. We allow S to be a single point, in which case $w(S) = 0$ and $\#(S) = 1$ or 0.

The **Root Isolation Problem** for a square S is to find a set of $\#(S)$ pairwise disjoint squares containing all the roots in S.

A **probe** in this setting is defined by a half-open square S and the result of the probe is the number of roots in the four smaller squares, S_i $(i = 1, 2, 3, 4)$, obtained by the segments joining the mid-points of the edges of S. Note that each of these smaller squares are half-open and they partition S. In this paper we do not discuss the implementation of this probe model, though similar implementations can be found in [Pin76, Wil78, Yap00, Pan97].

A probe S is called a **splitting probe** if either the centre of the square C is a root or $\#(S) \neq \#(S_i)$ for any $i = 1, \ldots, 4$; otherwise, we call it a **non-splitting probe**.

Suppose S contains k roots $\alpha_1, \ldots, \alpha_k$ in $\mathbb{C}$. For every root α_i inside S we define the following four pairs:

- $\sigma_{i,NE} = (\alpha_i, \alpha_{NE(i)})$, where $\alpha_{NE(i)}$ is the nearest root or corner of S lying north-east of α_i.
- $\sigma_{i,NW} = (\alpha_i, \alpha_{NW(i)})$, where $\alpha_{NW(i)}$ is the nearest root or corner of S lying north-west of α_i.
- $\sigma_{i,SE} = (\alpha_i, \alpha_{SE(i)})$, where $\alpha_{SE(i)}$ is the nearest root or corner of S lying south-east of α_i.
- $\sigma_{i,SW} = (\alpha_i, \alpha_{SW(i)})$, where $\alpha_{SW(i)}$ is the nearest root or corner of S lying south-west of α_i.

These pairs certainly exist; however, they may not be unique. We view these pairs as directed edges coming out from α_i. We also insist that these edges are never horizontal or vertical. This ensures that the four edges issuing out of each root are distinct (so our graph is simple). The set of edges in the directed graph so obtained for S will be represented as $\overline{\sigma}(S)$. By an **outer edge** of $\overline{\sigma}(S)$ we mean an edge connecting a root with a vertex of S; the remaining edges will be called **inner edges** and their set will be denoted by $I(S)$. For each $\sigma' \in \overline{\sigma}(S)$, let $w(\sigma')$ denote the Euclidean length of the edge σ'. We will write "$\sigma_{i,*}$" for any of the four edges coming out of α_i. Figure 1(a) illustrates the graph $\overline{\sigma}(S)$, where the dots represent the roots in S.

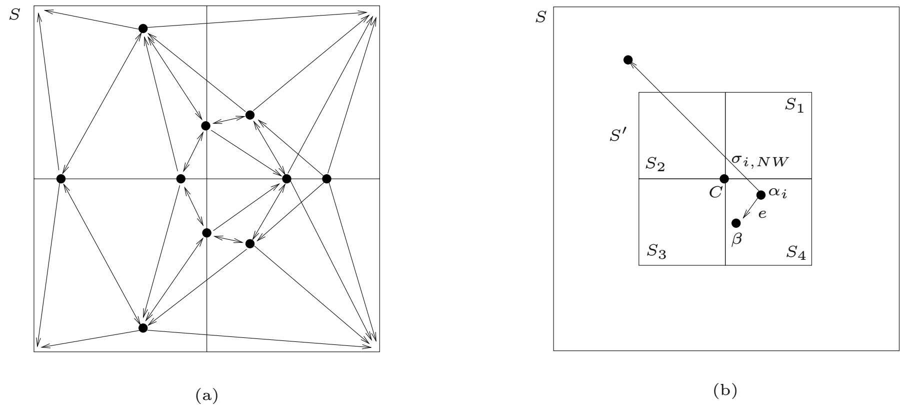


FIGURE 1. (a) The directed graph $\overline{\sigma}(S)$. (b) Charging scheme probe S'

Similar to Sect. 4 we define a threshold parameter $\tau > 0$ and define an edge $\sigma_{i,*}$ to be big or small depending on whether $w(\sigma_{i,*}) \geq \tau$ or $w(\sigma_{i,*}) < \tau$. We extend the definitions of (4.2) to this context,

$$\begin{aligned} \Delta^+ = \Delta^+(S) &:= \sum_{\sigma' \in \overline{\sigma}(S):\, \sigma' \text{ is big}} (1 + \lg(w(\sigma')/\tau)), \\ \Delta^- = \Delta^-(S) &:= \sum_{\sigma' \in \overline{\sigma}(S) \cap I(S):\, \sigma' \text{ is small}} (1 - \lg(w(\sigma')/\tau)). \end{aligned} \tag{6.1}$$

Let $\Delta(S) := \Delta^+ + \Delta^-$.

Analogous to Theorem 5 we show the following:

Theorem 9. *Given a square S and $k = \#(S)$, we can solve the root isolation problem in at most $1.5k - 1 + \Delta(S)$ probes.*

Proof. The algorithm based upon our probe model performs a four-way search. If $\#(S) \leq 1$ then we are done. Otherwise, initialize a queue containing S. In the general step, we extract a square S' from the queue and probe it. The consequence of the probe is the quadruple $(\#(S_1), \#(S_2), \#(S_3), \#(S_4))$ where S_i's are as shown in Figure 1(b). We enqueue S_i if $\#(S_i) > 1$, output it if $\#(S_i) = 1$, and discard it otherwise. The algorithm terminates when the queue is empty.

It is clear that there can be at most $k-1$ splitting probes. We now devise a charging scheme, analogous to what was done earlier, to account for the non-splitting probes. The analogy between the two charging schemes is the following: In Sect. 4 the charging of big segments was counting the number of intervals whose midpoint is contained within that segment; now the charging counts the number of squares that are sufficiently large and whose edges intersect the segment. The charging of small segments, in the same section, was counting the number of intervals that are sufficiently small and which contain the segment; now we count the number of squares in place of intervals.

Consider a non-splitting probe S' with center C. Since this probe is non-splitting all the roots inside S' must lie in one of S_i, $i = 1, 2, 3, 4$; suppose they lie in the square S_4 south-east of C. Let α_i be a root nearest to C. Thus the edge $\sigma_{i,NW}$ must intersect either the top or the left edge of S' and hence

$$w(\sigma_{i,NW} \cap S') \geq w(S_4) = \frac{1}{2} w(S'). \tag{6.2}$$

Furthermore, since S_4 is not an isolating square there is another root β inside S_4 such that one of the edges from α_i, apart from the north-west one, points to β; denote this edge by e. Then

$$w(e) \leq \sqrt{2} w(S_4) = w(S')/\sqrt{2}. \tag{6.3}$$

These notations are illustrated in Figure 1(b).

Now we describe our charging scheme:

1. If $w(S') \geq 2\tau$ then we charge the probe S' to the corresponding $\sigma_{i,NW}$. This means $\sigma' := \sigma_{i,NW}$ is big.
2. If $w(S') < \sqrt{2}\tau$ then we charge S' to the edge e. This means $\sigma' := e$ is small and cannot be an outer edge.
3. If $\sqrt{2}\tau \leq w(S') < 2\tau$ then we count an additional charge. However, there are at most $k/2$ such charges, since S' is not an isolating square and the next probe S_4 is such that $w(S_4) < \sqrt{2}\tau$.

Thus we have the following cases:

Case (A): σ' is small. Suppose $S_1, \ldots, S_\ell$ are the probes charged to σ'. Assume $S_\ell \subset S_{\ell-1} \subset \cdots \subset S_1$ and that σ' is in S_ℓ. Thus from (6.3) we have

$$\sqrt{2} w(\sigma') \leq w(S_\ell) \leq 2^{-1} w(S_{\ell-1}) \leq \cdots \leq 2^{-\ell+1} w(S_1) < 2^{-\ell+1}\sqrt{2}\tau,$$

and hence $\ell < 1 - \lg(w(\sigma')/\tau)$.

Case (B): σ' is big. Let $S_1, S_2, \ldots, S_\ell$ be all the probes charge to σ', such that $S_\ell \subset S_{\ell-1} \subset \cdots \subset S_2 \subset S_1$. Then from (6.2) we have

$$w(\sigma') \geq w(\sigma' \cap S_1) \geq \frac{1}{2} w(S_1) \geq w(S_2) \geq \cdots \geq 2^{\ell-2} w(S_\ell) \geq 2^{\ell-1}\tau,$$

and hence $\ell \leq 1 + \lg(w(\sigma')/\tau)$. □

7. Complexity of Complex Root Isolation

To bound the complexity of isolating all the roots of a degree d square-free polynomial $A(X)$ we need to bound the number of probes to achieve this. Let S be a half-open square containing all the roots of A; from Cauchy's bound we know that $w(S) \leq 2(1 + \|A\|_\infty)$. According to Theorem 9 we need to bound $\Delta^+(S)$ and $\Delta^-(S)$ for $\tau = 1$.

Lemma 10. *Let $A(X)$ be a degree d square-free integer polynomial with $L-$bit coefficients. Then we have $\Delta^-(S) = O(dL + d \lg d)$ and $\Delta^+(S) = O(dL)$.*

Proof. Suppose $\alpha_1, \ldots, \alpha_d$ are the distinct roots of $A(X)$ in S. We consider the upper bounds for $\Delta^-(S)$ and $\Delta^+(S)$ in turn:
Bound on $\Delta^-(S)$. For a root α_i let $\alpha_{N(i)}$ denote the nearest root to it. Consider the directed graph $G = (V, E)$ obtained from the pairs $(\alpha_i, \alpha_{N(i)})$. It is not hard to see that the cycles in G have the property that all the edges have the same length. Thus we can break any cycle of length greater than two to a set of cycles of length less than or equal to two. Now construct two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ from G where G_2 contains one of the edges from each cycle in G and G_1 contains all the remaining edges of G; thus $E = E_1 \cup E_2$. Both G_1 and G_2 are DAG's. Moreover, the vertices in G_2 have in-degree at-most one. We re-orient the edge $(\alpha, \beta) \in E_2$ such that $|\alpha| \leq |\beta|$; it is clear that this re-orientation does not affect the in-degree of the vertices. Now we apply Theorem 3 to G_2 to obtain:

$$\prod_{(\alpha,\beta)\in E_2} |\alpha - \beta| \geq \sqrt{|\mathrm{disc}(A)|} \cdot \mathrm{M}(A)^{-(d-1)} \cdot (d/\sqrt{3})^{-d} \cdot d^{-d/2}. \tag{7.1}$$

The in-degree of the vertices of G_1 can be at most six since any root of $A(X)$ can have at most six neighboring roots that are all closest to it and to each other. Now we re-orient the edge $(\alpha, \beta) \in E_1$ such that $|\alpha| \leq |\beta|$. As a result of this re-orientation the in-degree of any vertex in G_1 is bounded by three. The reason is that any circle that passes through the centre of the hexagon can contain at most three vertices since the vertices are diametrically opposite each other. Thus from Remark 3.1 we have:

$$\prod_{(\alpha,\beta)\in E_1} |\alpha - \beta| \geq |\mathrm{disc}(A)|^{1.5} \cdot \mathrm{M}(A)^{-(3d-3)} \cdot (d/\sqrt{3})^{-3d} \cdot d^{-1.5d}. \tag{7.2}$$

Combining this with (7.1) we have:

$$\prod_{(\alpha,\beta)\in E} |\alpha - \beta| \geq |\mathrm{disc}(A)|^{2} \cdot \mathrm{M}(A)^{-(4d-4)} \cdot (d/\sqrt{3})^{-4d} \cdot d^{-2d}. \tag{7.3}$$

We can partition the set of roots in V into four different types: R_i, $i = 1, 2, 3, 4$, be the set of roots which have exactly i inner edges where the edge $(\alpha, \beta) \in E$ is oriented such that $|\alpha| \leq |\beta|$. Then an upper bound on $\Delta^-(S)$ translates into a lower bound on $\prod w(\sigma')$, where σ' are small inner edges, but this product is equal to the product of the small inner edges corresponding to each root in R_i, $i = 1, 2, 3, 4$. This latter product is clearly greater than

$$\prod_{\alpha_i \in R_1} |\alpha_i - \alpha_{N(i)}| \cdot \prod_{\alpha_i \in R_2} |\alpha_i - \alpha_{N(i)}|^2 \cdot \prod_{\alpha_i \in R_3} |\alpha_i - \alpha_{N(i)}|^3 \cdot \prod_{\alpha_i \in R_4} |\alpha_i - \alpha_{N(i)}|^4,$$

since each inner edge is at least the distance to the nearest root. From (7.3) and (6.1), we thus have

$$\Delta^-(S) = O(dL + d \lg d),$$

since $\lg \mathrm{M}(A) \le L + 0.5 \lg(d+1)$.

Bound on $\Delta^+(S)$. Since S contains all the roots, we know $w(\sigma_{i,*})$, $i = 1, \ldots, d$, is less than the diagonal of S, which instead is less than 2^{L+3}. Thus for any root α_i we have

$$w(\sigma_{i,NW}) w(\sigma_{i,NE}) w(\sigma_{i,SW}) w(\sigma_{i,SE}) \le 2^{4L+12},$$

and from (6.1) we have

$$\begin{aligned} \Delta^+(S) &\le d + \textstyle\sum_{\sigma' \in \overline{\sigma}(S) : \sigma' \text{ is big}} \lg w(\sigma') \\ &\le 4dL + 13d, \end{aligned}$$

which clearly is $O(dL)$.

□

The bounds from the above lemma combined with Theorem 9 give the following:

Theorem 11. *Let $A(X)$ be a square-free integer polynomial of degree d and $L-$bit coefficients. Then the number of probes to isolate all the roots of $A(X)$ are at most $O(dL + d \lg d)$.*

Recall the distinction between the Sturm based approach for the case of real roots and complex roots. Since in the latter one has to compute a new Sturm sequence at each probe, this can increase the bit size of the coefficients of these polynomials by one, so that the final probe can possibly involve polynomials having bit size of the coefficients $\widetilde{O}(dL)$. The evaluation of the Sturm sequence corresponding to these polynomials can take $\widetilde{O}(d^4L^2)$, and hence the total complexity to isolate all roots of a degree d square-free integer polynomial with L-bit coefficients can potentially be $\widetilde{O}(d^5L^3)$. Hence Theorem 11 seems to be of theoretical interest only.

8. Conclusion

The two contributions of this paper are (1) a simplified approach for achieving the best known complexity bound in evaluating Sturm sequences, and (2) a new probe complexity bound for isolating complex roots. The common theme in these two results is the use of amortization arguments.

References

[CL83] G.E. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G. E. Collins and R. Loos, editors, *Computer Algebra*, pages 83–94. Springer-Verlag, 2nd edition, 1983.

[CLRS01] T.H. Corman, C.E. Leiserson, R.L. Rivest and C. Stein. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts and New York, second edition, 2001.

[Dav85] J.H. Davenport. Computer algebra for cylindrical algebraic decomposition. Technical report, The Royal Institute of Technology, Dept. of Numerical Analysis and Computing Science, S-100 44, Stockholm, Sweden, 1985. Reprinted as: Tech. Report 88-10, School of Mathematical Sciences, Univ. of Bath, Claverton Down, bath BA2 7AY, England.

[ESY06] A. Eigenwillig, V. Sharma and Chee Yap. Almost tight complexity bounds for the Descartes method. In *Proc. Int'l Symp. Symbolic and Algebraic Computation (ISSAC'06)*, 2006. To appear. Genova, Italy. Jul 9-12, 2006.

[Joh98] J.R. Johnson. Algorithms for polynomial real root isolation. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and monographs in Symbolic Computation, pages 269–299. Springer, 1998.

[KS94] M.-H. Kim and S. Sutherland. Polynomial root-finding algorithms and branched covers. *SIAM J. Computing*, 23:415–436, 1994.

[LR01] T. Lickteig and M.-F. Roy. Sylvester-Habicht sequences and fast Cauchy index computation. *J. of Symbolic Computation*, 31:315–341, 2001.

[Mc99] M. Mignotte and D. Ştefănescu. *Polynomials: An Algorithmic Approach.* Springer, 1999.

[Mil92] P.S. Milne. On the solutions of a set of polynomial equations. In B. R. Donald, D. Kapur, and J. L. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*, pages 89–102. Academic Press, London, 1992.

[NR96] C.A. Neff and J.H. Reif. An efficient algorithm for the complex roots problem. *J.Complexity*, 12(3):81–115, 1996.

[Pan96] V.Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers Mathematics and Applications*, 31(12):97–138, 1996.

[Pan97] V.Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Review*, 39(2):187–220, 1997.

[Ped90] P. Pedersen. Counting real zeroes. Technical Report 243, Courant Institute of Mathematical Sci., Robotics Lab., New York Univ., 1990. PhD Thesis, Courant Institute, New York University.

[Pin76] J.R. Pinkert. An exact method for finding the roots of a complex polynomial. *ACM Trans. on Math. Software*, 2:351–363, 1976.

[Rei97] D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC 97*, pages 233–240, 1997. Maui, Hawaii.

[Ren87] J. Renegar. On the worst-case arithmetic complexity of approximating zeros of polynomials. *Journal of Complexity*, 3:90–113, 1987.

[Sch82] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity, 1982. Manuscript, Department of Mathematics, University of Tübingen.

[Str83] V. Strassen. The computational complexity of continued fractions. *SIAM J. Computing*, 12:1–27, 1983.

[Wil78] H.S. Wilf. A global bisection algorithm for computing the zeros of polynomials in the complex plane. *Journal of the ACM*, 25:415–420, 1978.

[Yap00] C.K. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, 2000.

Zilin Du, Vikram Sharma and Chee K. Yap
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
e-mail: zilin@cs.nyu.edu
sharma@cs.nyu.edu
yap@cs.nyu.edu

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 131–147
© 2007 Birkhäuser Verlag Basel/Switzerland

An Algorithm of Real Root Isolation for Polynomial Systems with Applications to the Construction of Limit Cycles

Zhengyi Lu, Bi He, Yong Luo and Lu Pan

Abstract. By combining Wu's method, polynomial real root isolation and the evaluation of maximal and minimal polynomials, an algorithm for real root isolation of multivariate polynomials is proposed. Several examples from the literature are presented to illustrate the proposed algorithm.

Keywords. Wu's method, max-min polynomial, real root isolation.

1. Introduction

Solving systems of (symbolic or numerical) polynomial equations in several variables has been a very important topic of theoretical oriented research. Commonly used algorithmic methods for solving systems of polynomial equations include the characteristic set method introduced by Wu [20, 21], Gröbner basis method by Buchberger [1], and cylindrical algebraic decomposition method by Collins [4]. Numerical calculation methods include Newton method [18], homotopy method [9], and eigenvalue method [7].

A different approach for isolating the real roots of a univariate integral polynomial based on Sturm sequence, the derivative sequence and Descartes' rule of sign is widely used [8].

A natural questions is: Can the real root isolation algorithm be extended to the multivariate polynomial case?

In this paper, based on the characteristic set method, real root isolation and the evaluation for maximal and minimal polynomials, we propose an algorithm for isolating real roots of multivariate integral polynomial systems.

Partially supported by a National Key Basic Research Project of China (Granted no. 1998030600, 2004CB31800) and the National Natural Science Foundation of China (Granted no. 10371090) and Zhejiang Province Natural Science Foundation (Granted no. M103043).

This algorithm is an extension of a real root isolation algorithm for univariate integral polynomial. It results in a higher-dimensional isolated interval for each isolated real root [12, 13]. Based on this algorithm for multivariate polynomial systems, the stability analysis and the construction of small amplitude limit cycles for some differential polynomial systems are considered [12, 14, 15, 16].

Section 2 contains an introduction to Wu's method and the algorithm for real root isolation of multivariate polynomial systems. In Section 3, several examples from the literature are presented to illustrate the proposed algorithm. In Section 4, the construction of limit cycles for differential systems are given.

2. Wu's Method and Real Root Isolation

To prove the main result, we use the principle of the characteristic sets method which was introduced by Ritt [17] in the context of his work on differential algebra and has been considerably developed by Wu [20, 21]. The great success of theorem proving has stimulated the renewed interest in the characteristic sets method. To limit the space, we give here only the basic principle, i.e., the well ordering principle, and illustrate how this principle works.

Let $PS = \{f_1(x_1, \ldots, x_n), \ldots, f_s(x_1, \ldots, x_n)\}$ be any finite set of polynomials in n ordered variables $x_1 \prec \cdots \prec x_n$ with coefficients in certain basic field of characteristic 0, for instance, the field Q of rational numbers. We designate the complete set of zeros of the polynomials in PS by $\mathrm{Zero}(PS)$. If G is any other non-zero polynomial, the subset of $\mathrm{Zero}(PS)$ for which $G \neq 0$ will be denoted by $\mathrm{Zero}(PS/G)$. The following is the basic principle of the characteristic sets method [20, 21, 22].

Well Ordering Principle. Given a set PS of polynomials, one can compute by an algorithmic method another set CS of polynomials, called the *characteristic set* of PS, of the triangular form

$$CS: \quad \begin{cases} c_1(u_1, \ldots, u_d; y_1), \\ c_2(u_1, \ldots, u_d; y_1, y_2), \\ \qquad \cdots\cdots \\ c_r(u_1, \ldots, u_d; y_1, \ldots, y_r), \end{cases}$$

such that

$$\mathrm{Zero}(CS/J) \subset \mathrm{Zero}(PS) \subset \mathrm{Zero}(CS), \tag{2.1}$$

$$\mathrm{Zero}(PS) = \mathrm{Zero}(CS/J) \cup \bigcup_i \mathrm{Zero}(PS_i), \tag{2.2}$$

where $u_1, \ldots, u_d; y_1, \ldots, y_r$ $(d+r = n)$ is a rearrangement of $x_1, \ldots, x_n$, $J = \prod_i I_i$, I_i is the leading coefficient of c_i as polynomial in y_i, called the *initial* of c_i, and $PS_i = PS \cup \{I_i\}$.

The algorithm for triangularizing the polynomial set proceeds basically by successive pseudo-division of polynomials. From (2.1) and (2.2) the relation between the zeros of PS and CS is clear: any zero of PS is a zero of CS and, conversely, any zero of CS for which none of the initials vanishes is also a zero of PS. Therefore, under the condition that all the initials are not equal to 0, both PS and CS have the same zero set. For those zeros of PS making the vanishing of some initial I_i, we may consider further the enlarged polynomial set PS_i obtained from PS by adjoining I_i to it as required. Furthermore, in proceeding with each PS_i like PS by the same principle we would finally obtain a zero decomposition of the form

$$\mathrm{Zero}(PS) = \bigcup_i \mathrm{Zero}(CS_i/J_i), \tag{2.3}$$

in which CS_i is of triangular form as CS and J_i is the product of initials of the polynomials in CS_i for each i.

Equation (2.3) is called the Zero Decomposition Theorem [22]. Furthermore, by using any algorithm of polynomial factorization over algebraic extended fields, an ascending set CS can be decomposed into irreducible ascending sets. Hence, Wu obtains a variety decomposition theorem [22].

Variety Decomposition Theorem. For any given set PS of polynomials, there is an algorithm which computes a finite number of irreducible ascending sets IRR_k in a finite number of steps, such that

$$\mathrm{Zero}(PS) = \bigcup_k \mathrm{Zero}(IRR_k/J_k).$$

Moreover, redundant components can be removed by mere computations so that the union will become a non-contractible one.

Remark. Since the real roots of PS can be decomposed as the union of real roots of IRR_k, we only need to consider those components for which the irreducible ascending sets have isolated real roots. That is, the algebraic varieties associated to the irreducible ascending sets are of dimension zero. In such a case,

$$\mathrm{Zero}(IRR_k/J_k) = \mathrm{Zero}(IRR_k).$$

In the following discussions, we only consider the real root isolation algorithm in the positive cone R^n_+, since a transformation $x_i \to -x_i$ will change the negative real roots into the positive ones in R^n_+.

Now for any n-variate polynomial $f(x) = f(x_1, \ldots, x_n)$ in R^+_n, we denote the summation of the positive terms in $f(x)$ by $f^+(x_1, x_2, \ldots, x_n)$ and that of the negative terms in $f(x)$ by $f^-(x_1, x_2, \ldots, x_n)$. Clearly, $f = f^+ + f^-$.

The following theorems are direct results of the definition of f^+ and f^-.

Theorem 2.1. *If* $x_i \in [a, b]$ $(1 \le i \le n,\ 0 < a \le b)$, *then*

$$\begin{aligned} & f^+(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n) + f^-(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n) \\ \le\ & f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \\ \le\ & f^+(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n) + f^-(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n). \end{aligned}$$

Note that both f^+ and f^- are monotone in R_+^n. More general, we have:

Theorem 2.2. *If* $x_{i_j} \in [a_{i_j}, b_{i_j}]$, $1 \le i_j \le n$ $(j = 1, \dots, k)$, $0 < a_{i_j} \le b_{i_j}$, *then*

$$\begin{aligned} & f^+(x_1, \dots, a_{i_1}, \dots, a_{i_k}, \dots, x_n) + f^-(x_1, \dots, b_{i_1}, \dots, b_{i_k}, \dots, x_n) \\ \le\ & f(x_1, \dots, x_{i_1}, \dots, x_{i_k}, \dots, x_n) \\ \le\ & f^+(x_1, \dots, b_{i_1}, \dots, b_{i_k}, \dots, x_n) + f^-(x_1, \dots, a_{i_1}, \dots, a_{i_k}, \dots, x_n). \end{aligned}$$

In Theorem 2.1 (resp. Theorem 2.2) the identities hold if and only if $a = b$ (resp. $a_{i_j} = b_{i_j}$).

By Theorem 2.2, when considering all the variables $x_1, \dots, x_n$ in R_n^+, we can get the following estimation:

Theorem 2.3. *For given constants* $0 < a_i \le b_i$ $(i = 1, \dots, n)$,

(1) *if* $f^+(a_1, a_2, \dots, a_n) + f^-(b_1, b_2, \dots, b_n) > 0$, *then for* $x_i \in [a_i, b_i]$ $(i = 1, \dots, n)$,

$$f(x_1, x_2, \dots, x_n) > 0;$$

(2) *if* $f^+(b_1, b_2, \dots, b_n) + f^-(a_1, a_2, \dots, a_n) < 0$, *then for* $x_i \in [a_i, b_i]$ $(i = 1, \dots, n)$,

$$f(x_1, x_2, \dots, x_n) < 0.$$

The command *realroot* in Maple [2] can be used to isolate the real roots of a univariate polynomial with integral coefficients. To isolate the real roots of a given polynomial $g(x)$ with intervals of length less than or equal to ac, we can take *realroot*$(g(x), ac)$ which gives $[[a_1, b_1], \dots, [a_k, b_k]]$. Here, k is the exact number of distinct real roots of $g(x)$. Namely, in each interval $[a_i, b_i]$, there is one and only one real root of $g(x)$ and $b_i < a_{i+1}$ for all i. Furthermore, a_i and b_i have the same sign. This sequence of disjoint intervals with rational endpoints is called the *real root isolation intervals* of $g(x)$.

Now, we illustrate how to obtain the real root isolation domains of a triangular polynomial system $\{g_1(x), g_2(x, y)\}$ by *realroot* algorithm and the max-min polynomials.

The following notations and theorems will be used in the algorithm.

Definition 2.4. Suppose that $[a_1, b_1]$ is one of the real root isolation intervals of $g_1(x)$. The maximal and minimal polynomials of $g_2(x, y)$ on $[a_1, b_1]$ is defined by $\overline{g}_2(y)$ and $\underline{g}_2(y)$, respectively:

$$\overline{g}_2(y) = g_2^+(b_1, y) + g_2^-(a_1, y), \quad \underline{g}_2(y) = g_2^+(a_1, y) + g_2^-(b_1, y).$$

Clearly, if $\overline{x} \in [a_1, b_1]$ is a real root of $g_1(x)$, then $\underline{g}_2(y) \le g_2(\overline{x}, y) \le \overline{g}_2(y)$.

Theorem 2.5. *For the above* $\overline{g}_2(y)$, $\underline{g}_2(y)$, *there exists an interval with width* ac_1 *such that when* $b_1 - a_1 < ac_1$, *we have:*

(1) *the coefficients of the the leading term of both* $\overline{g}_2(y)$ *and* $\underline{g}_2(y)$ *have the same sign;*

(2) *the number of the real roots for both* $\overline{g}_2(y)$ *and* $\underline{g}_2(y)$ *are the same.*

Proof. Note that $\underline{g}_2(y) \to g_2(\bar{x}, y)$ and $\overline{g}_2(y) \to g_2(\bar{x}, y)$, if $a_1 \to \bar{x}$ and $b_1 \to \bar{x}$. Furthermore, $g_2(\bar{x}, y)$ is a squarefree polynomial in y; therefore, a perturbation of $\bar{x}$ will not change the properties of $g_2(\bar{x}, y)$: the changed polynomial will keep the sign of the leading coefficient as well as the number of real roots. □

Theorem 2.6. *Suppose that* $a \neq b$*; then there is an* ac_2 *such that the realroot command with accuracy (the length of isolating interval)* ac_2 *will lead to the sequence of real root isolation intervals of* $\overline{g}_2(y)$ *and* $\underline{g}_2(y)$*, respectively, as follows:*

$$L_1 := [[c_{11}, d_{11}], \ldots, [c_{1m}, d_{1m}]],$$

$$L_2 := [[c_{21}, d_{21}], \ldots, [c_{2m}, d_{2m}]].$$

Here, $[c_{1i}, d_{1i}] \bigcap [c_{2j}, d_{2j}] = \emptyset$ $(i, j = 1, \ldots, m)$. *Namely, we can get a total list of all the intervals in* L_1 *and* L_2.

Proof. In fact, the minimum root separation of an integral polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ is determined completely by its coefficients a_i $(i = 1, \ldots, n)$; see [5]. Suppose that $y_1, y_2, \ldots, y_k$ $(k \geq 2)$ are all the distinct real roots of $f(x)$; then

$$\min_{i,j \in \{1,\ldots,k\}} |y_i - y_j| > \frac{1}{n^{n+1}(1 + \sum_{l=0}^{n} |a_l|)}.$$

Now we consider the polynomial $g_{12}(y) = \overline{g}_2(y)\underline{g}_2(y)$ which is the product of $\overline{g}_2(y)$ and $\underline{g}_2(y)$. Take ac_2 as the minimum root separation of g_{12}; then Theorem 2.5 leads to this theorem. □

Remark 1. In a concrete manipulation, the accuracy ac_2 is much larger than the minimum root separation length.

Remark 2. When $a = b$, $\bar{x}$ is the rational solution of $g_(x)$. Hence, $\overline{g}_2(y) = \underline{g}_2(y) = g_2(\bar{x}, y)$. That is, $L_1 = L_2$ in Theorem 2.6.

Definition 2.7. The sequences of the real root isolation intervals of $\overline{g}_2(y)$ and $\underline{g}_2(y)$ are called *matchable* with respect to ac, if

(1) the sequences of the real root isolation intervals L_1 and L_2 take one of the following forms:

(i) 1, 2, 2, 1, 1, 2, 2, 1, ..., 1, 2, 2, 1;
(ii) 1, 2, 2, 1, 1, 2, 2, 1, ..., 1, 2, 2, 1, 1, 2;
(iii) 2, 1, 1, 2, 2, 1, 1, 2, ..., 2, 1, 1, 2;
(iv) 2, 1, 1, 2, 2, 1, 1, 2, ..., 2, 1, 1, 2, 2, 1

(here “1” denotes an interval belonging to L_1, and “2” to L_2),

(2) in each interval of L_1 and L_2, the considered integral polynomial $g_2(\bar{x}, y)$ is monotone.

Now we consider the triangular system$\{g_1(x),\ g_2(x, y)\}$.

Theorem 2.8. *Suppose that the real root $\bar{x}$ of $g_1(x)$ in $[a_1, b_1]$ makes the initial $J(\bar{x}) \neq 0$. If $g_2(\bar{x}, y)$ is squarefree with respect to y, then there must be a constant ac such that $\overline{g}_2(y)$ and $\underline{g}_2(y)$ are matchable with respect to ac.*

Proof. Since $g_2(\bar{x}, y)$ is squarefree with respect to y, it must pass through y-axes at each of its real roots. That is, $\overline{g}_2(y), \underline{g}_2(y) \to g_2(\bar{x}, y)$, as $a_1, b_1 \to \bar{x}$ ($ac \to 0$). Therefore, $\overline{g}_2(y)$ and $\underline{g}_2(y)$ also pass through the y-axes in the neighborhood of $g_2(\bar{x}, y)$.

Now consider the partial derivative $h(\bar{x}, y)$ of $g_2(\bar{x}, y)$ with respect to y. From Definition 2.4, we have

$$\underline{h}(y) \leq h(\bar{x}, y) \leq \overline{h}(y).$$

Clearly, for sufficiently small $|a_1 - b_1|$ (the length of the interval $[a_1, b_1]$), $\underline{h}(y)$ and $\overline{h}(y)$ have the same sign.

Hence, by Theorems 2.5 and 2.6, we can choose ac_1 as small as enough (i.e., $b_1 - a_1$ small enough) and take $ac = \min\{ac_1, ac_2\}$ such that $\overline{g}_2(y)$ and $\underline{g}_2(y)$ are matchable with respect to ac. □

The mrealroot algorithm for isolating the real roots of a set PS of polynomials can be described as follows (here, a three polynomial system will be considered). To guarantee the matchable condition in Theorem 2.8, we use Wu's method to decompose the real roots of PS to the union of those of a set of irreducible ascending sets.

Step 0. Consider system $PS = \{f_1(x, y, z), f_2(x, y, z), f_3(x, y, z)\}$. Using Wu's method (Variety Decomposition Theorem), we can get its irreducible components IRR_i, $i = 1, \ldots, k$. Take the components whose corresponding varieties are of dimension zero. Denote one IRR as $\{g_1(x), g_2(x, y), g_3(x, y, z)\}$.

Step 1. Using *realroot* command to get the sequence of isolation intervals of $g_1(x)$ (here, we just consider the positive interval), where the accuracy ac_1 is given by Theorem 2.5. Now we choose one interval, say $[a_1, b_1]$, to illustrate our method.

Step 2. Construct the maximal and minimal polynomials $\overline{g}_2(y)$ and $\underline{g}_2(y)$ of $g_2(x, y)$ with indeterminate y. By using the *realroot* algorithm, we can obtain the sequences of isolating real roots, L_1, L_2.

Step 3. Take ac and check if $\overline{g}_2(y)$ and $\underline{g}_2(y)$ are matchable with respect ac, i.e., if L_1 and L_2 take one of the forms (i), (ii), (iii) and (iv) in Definition 2.7. If the answer is positive, then go to step 4. Otherwise, divide ac by 2 and go back to step 1. By Theorem 2.8, after finitely many steps, L_1 and L_2 will be matchable.

Step 4. $\overline{g}_2(y)$ and $\underline{g}_2(y)$ are matchable with respect to ac. Now we match the interval as follows:

Take $a_{2i} = \min\{c_{1i}, c_{2i}\}$ and $b_{2i} = \max\{d_{1i}, d_{2i}\}$, $i = 1, \dots, m$; then $\{g_1(x), g_2(x, y)\}$ has just one real root in each $[a_1, b_1] \times [a_{2i}, b_{2i}]$ for $i = 1, \dots, m$.

Step 5. Suppose that $G = [a_1, b_1] \times [a_2, b_2]$ is a real root isolation interval of $\{g_1(x), g_2(x, y)\}$. We may construct the maximal and minimal polynomials $\overline{g}_3(z)$ and $\underline{g}_3(z)$ of $g_3(x, y, z)$ as follows:

$$\overline{g}_3(z) = g_3^+(b_1, b_2, z) + g_3^-(a_1, a_2, z),$$
$$\underline{g}_3(z) = g_3^+(a_1, a_2, z) + g_3^-(b_1, b_2, z).$$

Step 6. Similar to steps 2 and 4, we obtain the real root isolation interval $G = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ of $\{g_1(x), g_2(x, y), g_3(x, y, z)\}$.

Step 7. Since the initials of IRR are not zero at its real roots, we can have all the real root isolation intervals of

$$\{f_1(x, y, z), f_2(x, y, z), f_3(x, y, z)\}.$$

For a general n-variate polynomial system, we may describe the above seven-step algorithm (except step 0) as follows:

```
INPUT      $\{g_1(x_1), g_2(x_1, x_2), \dots, g_n(x_1, x_2, \dots, x_n)\}$;   $J(x_1, \dots, x_n)$
OUTPUT     the real root isolation intervals of
           $\{g_1(x_1), g_2(x_1, x_2), \dots, g_n(x_1, x_2, \dots, x_n)\}$;
           the signs of $J(x_1, \dots, x_n)$ at every isolating interval
BEGIN
    LABEL
    $G_1 := \{realroot(g_1(x_1)$, ac) intervals for positive real roots$\}$
    IF $G_1 = \emptyset$ THEN return $\emptyset$ END IF
    $k := 2$                                                   /* remark 3 */
    WHILE $k \le n$ DO
        $G_k = \emptyset$
        $g_k(x_1, \dots, x_k) = g_k^+(x_1, \dots x_k) + g_k^-(x_1, \dots, x_k)$
        FOR $p$ in $G_{k-1}$ DO
            $\underline{g}_k(x_k)$,   $\overline{g}_k(x_k)$                               /* remark 4 */
            $\underline{G}_k = \{realroot(\underline{g}_k(x_k)$, ac) intervals with positive endpoints$\}$
            $\overline{G}_k = \{realroot(\overline{g}_k(x_k)$, ac) intervals with positive endpoints$\}$
            flag = $Q_k(\underline{G}_k, \overline{G}_k)$                          /* remark 5 */
            IF (flag == false) GOTO LABEL
            $G_k = G_k$ union $Q_k(\underline{G}_k, \overline{G}_k)$
        END FOR
    $k := k + 1$
    END WHILE                                                  /* remark 6*/
END
```

Remark 3. k denotes the k-th unknown x_k for the k-th polynomial.

Remark 4. $\underline{g}_k(x_k)$ and $\overline{g}_k(x_k)$ are minimal and maximal polynomials of $g_k(x_1,\ldots,x_k)$, respectively.

Remark 5. Check if $\overline{g}_k(x_k)$ and $\underline{g}_k(x_k)$ are matchable with respect to ac. If the answer is positive, then finish the match to get the isolation intervals. Otherwise, choose $ac/2$ as new ac and return "false".

Remark 6. The result G_n is just the set of real root isolation intervals of

$$\{g_1(x_1), g_2(x_1,x_2),\ldots,g_n(x_1,x_2,\ldots,x_n)\}.$$

Based on the above algorithm, we construct a command *mrealroot* in Maple:

$$> mrealroot([g_1(x_1), g_2(x_1,x_2),\ldots,g_n(x_1,x_2,\ldots,x_n)],[x_1,x_2,\ldots,x_n], \\ c, \quad [h_1(x_1,x_2,\ldots,x_n),\ldots,h_m(x_1,x_2,\ldots,x_n)]);$$

where n, m are positive integers and the positive number c is the upper bound of the width of the intervals of the real roots. Here c is used to control the accuracy of the interval solutions to its exact ones. If c is too wide to fulfil the matchable condition in Theorem 2.8, it will be substituted by a smaller one automated according to the algorithm. Even if c is omitted, the most convenient width is used for each interval returned.

The output is:

$$\Big[\big[x_{11},x_{12}\big],\big[x_{21},x_{22}\big],\ldots,\big[x_{n1},x_{n2}\big],\underbrace{+,-,\widetilde{0},\ldots,-}_{m}\Big].$$

Here, "+" (resp. "−") denotes that a polynomial at the real root is positive (resp. negative), $\widetilde{0}$ means that the positiveness and negativeness are not determined (undecided). The above output means that there is a real root

$$(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n) \in \big[\big[x_{11},x_{12}\big] \times \big[x_{21},x_{22}\big] \times \cdots \times \big[\big[x_{n1},x_{n2}\big]\big]$$

such that $h_1(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n) > 0, h_2(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n) < 0, h_3(\overline{x}_1,\overline{x}_1,\ldots,\overline{x}_n)$ is undecided, ..., $h_m(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n) < 0$.

3. Application to Polynomial Systems

The first example is from [6].

Example P1.

$$\begin{aligned} f_1 &= x^5 + y^2 + z^2 - 4,\\ f_2 &= x^2 + 2y^2 - 5,\\ f_3 &= xz - 1. \end{aligned} \tag{3.1}$$

By using step 0 in the algorithm, (3.1) is transformed to the triangular form:

$$\begin{aligned} g_1 &= 2x^7 + 2 - 3x^2 - x^4,\\ g_2 &= x^2 + 2y^2 - 5,\\ g_3 &= xz - 1. \end{aligned}$$

Take the *mrealroot* command

$$> mrealroot([g_1, g_2, g_3], [x, y, z], \frac{1}{10})$$

we get

$$\Big[\Big[1,1\Big], \Big[\tfrac{11}{8}, \tfrac{23}{16}\Big], \Big[1,1\Big]\Big], \qquad \Big[\Big[\tfrac{13}{16}, \tfrac{7}{8}\Big], \Big[\tfrac{23}{16}, \tfrac{3}{2}\Big], \Big[\tfrac{9}{8}, \tfrac{5}{4}\Big]\Big],$$

$$\Big[\Big[1,1\Big], \Big[-\tfrac{23}{16}, -\tfrac{11}{8}\Big], \Big[1,1\Big]\Big], \quad \Big[\Big[-\tfrac{3}{4}, -\tfrac{11}{16}\Big], \Big[\tfrac{23}{16}, \tfrac{25}{16}\Big], \Big[-\tfrac{3}{2}, -\tfrac{21}{16}\Big]\Big],$$

$$\Big[\Big[\tfrac{13}{16}, \tfrac{7}{8}\Big], \Big[-\tfrac{3}{2}, -\tfrac{23}{16}\Big], \Big[\tfrac{9}{8}, \tfrac{5}{4}\Big]\Big], \quad \Big[\Big[-\tfrac{3}{4}, -\tfrac{11}{16}\Big], \Big[-\tfrac{25}{16}, -\tfrac{23}{16}\Big], \Big[-\tfrac{3}{2}, -\tfrac{21}{16}\Big]\Big].$$

This means that the polynomial system (3.1) has six real roots.

The second example is also from [6].

Example P2.

$$\begin{aligned} f_1 &= x^2 - 2xz + 5, \\ f_2 &= xy^2 + yz + 1, \\ f_3 &= 3y^2 - 8xz. \end{aligned} \tag{3.2}$$

The step 0 gives us from (3.2) the triangular form

$$\begin{aligned} g_1 &= y(320 + 1600y^4 - 240y^5 - 471y^6 + 36y^7 - 48y^2 + 36y^8), \\ g_2 &= -40y^2 + 3y^3 + 6y^4 + 8x, \\ g_3 &= -8yz - 8 - 40y^4 + 3y^5 + 6y^6. \end{aligned}$$

Applying *mrealroot*:

$$> mrealroot([g_1, g_2, g_3], [x, y, z], \frac{1}{10^5})$$

we obtain two real root isolation intervals:

$$\Big[\Big[-\tfrac{188613}{65536}, -\tfrac{377225}{131072}\Big], \Big[-\tfrac{72181}{65536}, -\tfrac{36063}{32768}\Big], \Big[-\tfrac{185017}{65536}, -\tfrac{369513}{131072}\Big]\Big],$$

$$\Big[\Big[-\tfrac{45}{16}, -\tfrac{368639}{131072}\Big], \Big[\tfrac{126525}{131072}, \tfrac{63315}{65536}\Big], \Big[\tfrac{50295}{16384}, \tfrac{100711}{32768}\Big]\Big].$$

Example P3 [5].

$$\begin{aligned} f_1 &= x^2 + 2y^2 - y - 2z, \\ f_2 &= x^2 - 8y^2 + 10z - 1, \\ f_3 &= x^2 - 7yz. \end{aligned} \tag{3.3}$$

The triangular form of (3.3) is

$$\begin{aligned} g_1 &= 2450x^6 - 1241x^4 + 196x^2 - 49, \\ g_2 &= 86x^2 + 35yx^2 - 77y - 14, \\ g_3 &= 175x^4 + 70x^2z - 76x^2 - 154z + 21, \end{aligned}$$

with initial $J = 5x^2 - 11$.

Now the command

$$> mrealroot([g_1, g_2, g_3], [x, y, z], \frac{1}{10^5}, [J])$$

gives out two real root isolation intervals:

$$\left[\left[\frac{85823}{131072}, \frac{1341}{2048}\right], \left[\frac{48355}{131072}, \frac{24179}{65536}\right], \left[\frac{10879}{65536}, \frac{10881}{65536}\right], -\right],$$

$$\left[\left[-\frac{1341}{2048}, -\frac{85823}{131072}\right], \left[\frac{48355}{131072}, \frac{24179}{65536}\right], \left[\frac{10879}{65536}, \frac{10881}{65536}\right], -\right].$$

Example P4 [5].

$$\begin{aligned} f_1 &= x^4 + y^4 - 1, \\ f_2 &= x^5y^2 - 4x^3y^3 + x^2y^5 - 1. \end{aligned} \tag{3.4}$$

The triangular form of (3.4) is as follows:

$$\begin{aligned} g_1 = {} & 16y^5 - 36y^9 + 30y^{10} - y^8 + 144y^{15} - 64y^{11} + 758y^{16} - 251y^{12} + 28y^{13} \\ & - 126y^{14} + 4y^{21} - 12y^{17} - 128y^{22} + 192y^{18} + 48y^{23} - 112y^{19} - 757y^{20} \\ & + 32y^{26} - 16y^{27} + 249y^{24} + 2y^{28} + 1, \\ g_2 = {} & 4x + y - 2y^5 + 2y^8x - 2y^4x + y^2 + 16y^3 - 24y^7 - xy^3 + 4y^{10}x - 3y^{11}x + 3y^7x + y^9 \\ & - 32xy^9 + 16y^{13}x - 4y^{14}x + y^{15}x - 8y^{15} + 16y^{11} + y^{16} - y^{12} + 16y^5x. \end{aligned}$$

The initial is

$$J = 4 + 2y^8 - 2y^4 - y^3 + 4y^{10} - 3y^{11} + 3y^7 - 32y^9 + 16y^{13} - 4y^{14} + y^{15} + 16y^5.$$

Taking the command

$$> mrealroot([g_1, g_2], [y, x], \frac{1}{10^5}, [J]);$$

we get four real roots:

$$\left[\left[-\frac{121139}{131072}, -\frac{60569}{65536}\right], \left[\frac{47213}{65536}, \frac{47299}{65536}\right], +\right], \quad \left[\left[\frac{94517}{131072}, \frac{47259}{65536}\right], \left[-\frac{60575}{65536}, -\frac{60563}{65536}\right], +\right],$$

$$\left[\left[-\frac{78307}{131072}, -\frac{39153}{65536}\right], \left[\frac{126675}{131072}, \frac{31673}{32768}\right], +\right], \quad \left[\left[\frac{126681}{131072}, \frac{63341}{65536}\right], \left[-\frac{19599}{32768}, -\frac{39111}{65536}\right], +\right].$$

In [4], Collins considered the following system of polynomials and found four solution points. By using the mrealroot algorithm, a PIII 550 computer gives out the four isolating real roots in less than 1.2 seconds.

Example P5.

$$\begin{aligned} f_1 &= -7xyz + 6yz - 14xz + 9z - 3xy - 12y - x + 1, \\ f_2 &= 2xyz - yz + 14z + 15xy + 14y - 15x, \\ f_3 &= -8xyz + 11yz - 12xz - 5z + 15xy + 2y + 10x - 14. \end{aligned} \tag{3.5}$$

The triangular form of (3.5) is as follows:

$$\begin{aligned}g_1 &= -311308988x - 7694683176x^2 + 4541529810x^3 + 8951356045x^4\\ &\quad - 4587245014x^5 - 4919456115x^6 + 3307892784 + 198993030x^7\\ &\quad + 174040200x^8,\\ g_2 &= 34405x^2 + 7530x^4y - 46172x^3y - 65274x^2y + 22815x^3 - 13680x^4\\ &\quad - 17640x + 44624xy - 16296 + 54084y,\\ g_3 &= 92250x^5 + 46110x^4z + 541875x^4 - 692403x^3z + 143625x^3 - 844151x^2z\\ &\quad - 886430x^2 + 639688xz - 319860x + 740880z + 27360x^5z + 228144,\end{aligned}$$

and the initial is

$$\begin{aligned}J &= -186017052192109889164x + 349875023661822013792x^2\\ &\quad - 186848402623007386128 + 337522017343330106770x^3\\ &\quad - 170675129780428490075x^4 - 155540642141382952602x^5\\ &\quad + 99439300041231137485x^6 + 59968621217751821890x^7.\end{aligned}$$

Call the mrealroot command:

$$> mrealroot([g_1, g_2, g_3], [x, y, z], \frac{1}{10^{10}}, [J]);$$

the computer shows four isolating real roots:

$$[[\tfrac{33573910397}{34359738368}, \tfrac{16786955199}{17179869184}], [\tfrac{82505394681}{8589934592}, \tfrac{330021582735}{34359738368}], [-\tfrac{96628390617}{8589934592}, -\tfrac{386513559595}{34359738368}], -1],$$
$$[[-\tfrac{8880815695}{8589934592}, -\tfrac{35523262779}{34359738368}], [-\tfrac{16638954157}{17179869184}, -\tfrac{33277908053}{34359738368}], [-\tfrac{1073569409}{1073741824}, -\tfrac{34354220805}{34359738368}], -1],$$
$$[[-\tfrac{7659282837}{8589934592}, -\tfrac{30637131347}{34359738368}], [\tfrac{614939185923}{34359738368}, \tfrac{614939235161}{34359738368}], [\tfrac{11783002047}{17179869184}, \tfrac{11783002915}{17179869184}], 1],$$
$$[[\tfrac{173688058101}{34359738368}, \tfrac{86844029051}{17179869184}], [-\tfrac{36761640685}{17179869184}, -\tfrac{36761640679}{17179869184}], [-\tfrac{1676409165625}{34359738368}, -\tfrac{1676409164939}{34359738368}], 1].$$

In all the above examples, the systems are considered in R^n. In fact, we may consider systems in C^n. In this case a transformation $x \to x_1 + ix_2$ shall be made, where x_1 is the real part and x_2 the imaginary part of x.

Example P6.

$$\begin{aligned}f_1 &= 2x^2 - xy + 4,\\ f_2 &= xy - 2y^2 + 4.\end{aligned} \tag{3.6}$$

After the transformation $x \to x_1 + ix_2$, $y \to y_1 + iy_2$, we have

$$\begin{aligned}f_1 &= p_{11} + ip_{12},\\ f_2 &= p_{21} + ip_{22},\end{aligned}$$

where

$$\begin{aligned}p_{11} &= 2x_1^2 + 4 - 2x_2^2 - x_1y_1 + x_2y_2,\\ p_{12} &= 4x_1x_2 - x_1y_2 - x_2y_1,\\ p_{21} &= x_1y_1 + 4 - x_2y_2 - 2y_1^2 + 2y_2^2,\\ p_{22} &= x_1y_2 + x_2y_1 - 4y_1y_2.\end{aligned} \tag{3.7}$$

By using Wu's method, (3.7) can be transformed to

$$\begin{aligned} g_1 &= 3y_1^4 - 6y_1^2 - 1, \\ g_2 &= 2y_1^3 - x_1y_1^2 - 4y_1 + x_1, \\ g_3 &= -y_1^2 + 2 + y_2^2, \\ g_4 &= y_1^2x_2 - 2y_1^2y_2 - x_2. \end{aligned} \tag{3.8}$$

The initial is $J = (y_1 - 1)(y_1 + 1)$. Taking the *mrealroot* command for (3.8):

$$> mrealroot([g_1, g_2, g_3, g_4], [y_1, x_1, y_2, x_2], \frac{1}{10^2}, [J]);$$

we obtain four pairs of real roots

$$\left[\left[\tfrac{187}{128}, \tfrac{47}{32}\right], \left[\tfrac{39}{128}, \tfrac{7}{16}\right], \left[\tfrac{23}{64}, \tfrac{51}{128}\right], \left[\tfrac{169}{128}, \tfrac{97}{64}\right], +\right],$$

$$\left[\left[\tfrac{187}{128}, \tfrac{47}{32}\right], \left[\tfrac{39}{128}, \tfrac{7}{16}\right], \left[-\tfrac{51}{128}, -\tfrac{23}{64}\right], \left[-\tfrac{97}{64}, -\tfrac{199}{128}\right], +\right],$$

$$\left[\left[-\tfrac{47}{32}, -\tfrac{187}{128}\right], \left[-\tfrac{7}{16}, -\tfrac{39}{128}\right], \left[\tfrac{23}{64}, \tfrac{51}{128}\right], \left[\tfrac{169}{128}, \tfrac{97}{64}\right], +\right],$$

$$\left[\left[-\tfrac{47}{32}, -\tfrac{187}{128}\right], \left[-\tfrac{7}{16}, -\tfrac{39}{128}\right], \left[-\tfrac{51}{128}, -\tfrac{23}{64}\right], \left[-\tfrac{97}{64}, -\tfrac{169}{128}\right], +\right].$$

Therefore, the corresponding four complex roots of (3.6) (with $x = (x_1, x_2)$, $y = (y_1, y_2)$) are

$$\left[\left[\tfrac{39}{128}, \tfrac{7}{16}\right] \times \left[\tfrac{169}{128}, \tfrac{97}{64}\right], \left[\tfrac{187}{128}, \tfrac{47}{32}\right] \times \left[\tfrac{23}{64}, \tfrac{51}{128}\right]\right],$$

$$\left[\left[\tfrac{39}{128}, \tfrac{7}{16}\right] \times \left[-\tfrac{97}{64}, -\tfrac{199}{128}\right], \left[\tfrac{187}{128}, \tfrac{47}{32}\right] \times \left[-\tfrac{51}{128}, -\tfrac{23}{64}\right]\right],$$

$$\left[\left[-\tfrac{7}{16}, -\tfrac{39}{128}\right] \times \left[\tfrac{169}{128}, \tfrac{97}{64}\right], \left[-\tfrac{47}{32}, -\tfrac{187}{128}\right] \times \left[\tfrac{23}{64}, \tfrac{51}{128}\right]\right],$$

$$\left[\left[-\tfrac{7}{16}, -\tfrac{39}{128}\right] \times \left[-\tfrac{97}{64}, -\tfrac{169}{128}\right], \left[-\tfrac{47}{32}, -\tfrac{187}{128}\right] \times \left[-\tfrac{51}{128}, -\tfrac{23}{64}\right]\right].$$

4. Application to Differential Systems

The problem of distinguishing between centers and foci and the construction of small amplitude limit cycles for polynomial differential systems has a tight connection with Hilbert's 16th problem. Based on the real root isolation algorithm, the construction of small amplitude limit cycles for differential polynomial systems was proposed in [11]. After the Liapunov constants are obtained for a system, the question for the estimation of the number of small amplitude limit cycles bifurcated from a fine focus becomes the following: can we isolate the real roots for the polynomial system of the first k Liapunov constants?

In this section, based on the method in [14] and the *mrealroot* algorithm, we give three examples for cubic systems.

In what follows, $L_j(i)$ denotes the j-th Liapunov constant (focal value) for the i-th example.

Example D1. In 1980, Coleman [3] proposed the conjecture that a Kolmogorov prey-predator system may have more than two stable limit cycles. In [13], this conjecture is confirmed. In that paper, the idea of *mrealroot* algorithm is proposed.

The constructed system is as follows:

$$\begin{aligned}\dot{x} &= x(-2 - a_0 + a_1 + a_0 x - 2a_1 x + y + a_1 x^2 + xy),\\ \dot{y} &= y(2 + a_2 - x - y - 2a_2 y + a_2 y^2).\end{aligned}$$

Here, $a_1 > 0, a_2 > 0$. Clearly, $(1,1)$ is a positive equilibrium of the system. By using the transformation

$$\overline{x} = x - 1, \qquad \overline{y} = y - 1,$$

the original system (we use x, y instead of $\overline{x}, \overline{y}$) is changed to

$$\begin{aligned}\dot{x} &= (x+1)(x + a_0 x + 2y + a_1 x^2 + xy),\\ \dot{y} &= (y+1)(-x - y + a_2 y^2).\end{aligned} \tag{4.1}$$

When $a_0 = 0$, $(0,0)$ is a focus. The first three focal values of (4.1) are

$$\begin{aligned}L_1(1) &= 3a_2 - a_2^2 + 2a_1 - 1 + 2a_1^2,\\ L_2(1) &= 451a_2^2 + 546a_2a_1 + 427a_1^2 - 392a_2^2a_1 - 328a_2^3 - 472a_1^3 + 46a_2^3a_1\\ &\quad - 92a_2a_1^3 + 176a_2^2a_1^2 - 876a_2a_1^2 + 74a_2^4 - 648a_1^4 - 238a_2 - 182a_1 + 44,\\ L_3(1) &= -889784a_2^4 - 28184 + 3433278a_2a_1^2 + 944669a_2^3 - 1652639a_2^3a_1\\ &\quad + 1918142a_1^3 - 5270694a_2a_1^3 - 5028128a_2^2a_1^2 - 3163812a_1^4\\ &\quad - 579519a_2^2 + 221080a_1 + 196690a_2 - 1112353a_2a_1 + 2063089a_2^2a_1\\ &\quad + 499258a_1a_2^4 + 135384a_1^6 + 763936a_2a_1^5 - 338976a_2^3a_1^3 + 308.\end{aligned}$$

We can obtain simpler and equivalent $\bar{L}_1, \bar{L}_2, \bar{L}_3$ with $\bar{L}_1 = L_1(1)$, $\bar{L}_3 = L_3(1)$ and

$$\bar{L}_2 = 638a_2^3 - 709a_2^4 - 30a_2 - 139a_2^2 + 120a_2^5 + 8a_2^6 + 3,$$

provided that the initial $I = 8a_2^3 - 51a_2^2 - 12a_2^2a_1 + 32a_2a_1 + 55a_2 - 19 - 14a_1 \neq 0$. Now, taking the commend

$$mrealroot([\bar{L}_1, \bar{L}_2], [a_2, a_1], 1/10^{10}, [\bar{L}_3, I]),$$

we obtain eight solutions:

$$\left[\left[\frac{688939015}{8589934592}, \frac{1377878031}{17179869184}\right], \qquad \left[\frac{5077638577}{17179869184}, \frac{5077638579}{17179869184}\right], \quad -, -\right],$$

$$\left[\left[\frac{688939015}{8589934592}, \frac{1377878031}{17179869184}\right], \qquad \left[-\frac{22257507763}{17179869184}, -\frac{22257507761}{17179869184}\right], \quad -, \widetilde{0}\right],$$

$$\left[\left[-\frac{2985891391}{17179869184}, -\frac{1492945695}{8589934592}\right], \qquad \left[\frac{8810195803}{17179869184}, \frac{8810195805}{17179869184}\right], \quad +, -\right],$$

$$\left[\left[-\frac{84633102747}{4294967296}, -\frac{338532410987}{17179869184}\right], \quad \left[-\frac{265975581777}{17179869184}, -\frac{265975581775}{17179869184}\right], \quad -, \widetilde{0}\right],$$

$$\left[\left[\frac{62789278049}{17179869184}, \frac{31394639025}{8589934592}\right], \qquad \left[\frac{7689817571}{8589934592}, \frac{15379635145}{17179869184}\right], \quad +, -\right],$$

$$\left[\left[-\frac{84633102747}{4294967296}, -\frac{338532410987}{17179869184}\right], \qquad \left[\frac{248795712591}{17179869184}, \frac{248795712593}{17179869184}\right], \quad +, -\right],$$

$$\left[\left[\frac{62789278049}{17179869184}, \frac{31394639025}{8589934592}\right], \qquad \left[-\frac{32559504329}{17179869184}, -\frac{16279752163}{8589934592}\right], \quad -, \widetilde{0}\right],$$

$$\left[\left[-\frac{2985891391}{17179869184}, -\frac{1492945695}{8589934592}\right], \qquad \left[-\frac{25990064989}{17179869184}, -\frac{25990064987}{17179869184}\right], \quad -, \widetilde{0}\right].$$

The first solution (with $a_1 > 0, a_2 > 0$) is what we need. In this case, the system can have three small amplitude limit cycles among which two are stable.

Example D2. In [10], the following cubic Kolmogorov system is considered:

$$\begin{aligned} \dot{x} &= x(x - 2y + 2)(Ax + y + B), \\ \dot{y} &= y(2x - y - 2)(Dx + y + C). \end{aligned} \tag{4.2}$$

Here $(2, 2)$ is a positive fixed point. With the transformation $\overline{x} = x - 2, \overline{y} = y - 2$, system (4.2) takes the form (here we use x, y instead of $\overline{x}, \overline{y}$)

$$\begin{aligned} \dot{x} &= (x + 2)(x - 2y)(A(x + 2) + y + 2 + B), \\ \dot{y} &= (y + 2)(2x - y)(D(x + 2) + y + 2 + C). \end{aligned} \tag{4.3}$$

To ensure $(0, 0)$ to be a center-focus form, we need $D = A+B/2-C/2$. Substituting it into (4.3), we have

$$\begin{aligned} \dot{x} &= (x + 2)(x - 2y)(A(x + 2) + y + 2 + B), \\ \dot{y} &= (y + 2)(2x - y)((A + B/2 - C/2)(x + 2) + y + 2 + C). \end{aligned} \tag{4.4}$$

At $(0, 0)$, the first four focal values of (4.4) are $L_1(2), L_2(2), L_3(2), L_4(2)$. When $C = -4$, $I_{21} = 2 + B + 2A \neq 0$, $I_{22} \neq 0$ and $b + 4 \neq 0$, we can have reduced focal values: $\bar{L}_1 = L_1(2) = 0$, $\bar{L}_4 = L_4(2)$ and

$$\begin{aligned} \bar{L}_2 &= -3B^2 + 4AB - 22B + 28A^2 + 56A, \\ \bar{L}_3 &= 5639949B^8 - 158890977B^7 - 9906667659B^6 - 118203801471B^5 \\ &\quad - 407483203554B^4 + 118362692448B^3 + 417384231264B^2, \end{aligned}$$

with

$$\begin{aligned} I_2 = {} & 18252402AB^3 + 309080730B^2A + 1053193848AB - 3997665B^4 \\ & - 132974583B^3 - 759844470B^2 + 403771368B. \end{aligned}$$

Taking the commend

$$mrealroot([\bar{L}_2, \bar{L}_3], [B, A], 1/10^{20}, [\bar{L}_4, I_2, B+4]),$$

we have four solutions

$$\begin{aligned} &\Bigg[\Bigg[-\frac{2573675109802661915841}{147573952589676412928}, -\frac{402136735906665924 35}{2305843009213693952}\Bigg], \\ &\qquad \Bigg[-\frac{1183730637891661183}{288230376151711744}, -\frac{3030350433002652628 47}{7378697629483820646 4}\Bigg], \quad +, +, -, -\Bigg], \\ &\Bigg[\Bigg[-\frac{290628590797434259035}{36893488147419103232}, -\frac{1162514363189737036139}{147573952589676412928}\Bigg], \\ &\qquad \Bigg[\frac{145450142956256898121}{73786976294838206464}, \frac{72725071478128449063}{36893488147419103232}\Bigg], \quad -, +, -, -\Bigg], \\ &\Bigg[\Bigg[-\frac{1278947606251465505551}{147573952589676412928}, -\frac{639473803125732752775}{73786976294838206464}\Bigg], \\ &\qquad \Bigg[\frac{80151834260523818841}{36893488147419103232}, \frac{20037958565130954711}{9223372036854775808}\Bigg], \quad -, +, -, -\Bigg], \\ &\Bigg[\Bigg[-\frac{1278947606251465505551}{147573952589676412928}, -\frac{639473803125732752775}{73786976294838206464}\Bigg], \\ &\qquad \Bigg[\frac{117330199710287679315}{147573952589676412928}, \frac{58665099855143839661}{73786976294838206464}\Bigg], \quad -, +, -, -\Bigg]. \end{aligned}$$

Clearly at all the four zeros, $\bar{L}_1$, $\bar{L}_2$, and $\bar{L}_3$ are independent with respect to $\bar{L}_4$; therefore there are four classes of values for A, B, C, D such that system (4.2) has four small amplitude limit cycles at $(0, 0)$.

Example D3. Consider the following system [19]

$$\begin{aligned} \dot{x} &= y + Bx^3 + (C-G)x^2y + (3D-H)xy^2 + Ey^3, \\ \dot{y} &= -x - Ax^3 - (3B+F)x^2y - (C+G)xy^2 - Dy^3. \end{aligned} \tag{4.5}$$

Clearly, $(0, 0)$ is an equilibrium. The first five focal values are $L_1(3), \ldots, L_5(3)$. Well ordering the focal values in the order of $H \prec F \prec A \prec G$, we obtain

$$\begin{aligned} L_1 &= -F - H, \\ L_2 &= 4DG + 2FG - FE + FA + 4BG, \\ L_3 &= Gp_{53}q_{51}I_{51}^{-1}, \\ L_4 &= -G^2p_{54}q_{51}I_{51}^{-1}, \\ L_5 &= (32EC + 48E^2 - 9D^2 + 30DB - 9B^2)(-9B^3 - 9B^2D + 32BC^2 + 144BCE \\ &\quad + 144BE^2 + 9BD^2 - 48DCE - 144DE^2 + 9D^3), \end{aligned}$$

where

$$\begin{aligned}
p_{53} &= 5A - 5E + 2G,\\
p_{54} &= 3G - 15E - 5C,\\
q_{51} &= (A^2D - ADC + BEA - BGA - BCA + 3DAG - DAE + 2B^3\\
&\quad - DGE + DCE - 2DCG + 2DG^2 - 2D^3 - 2BD^2 - BE^2 - 2BG^2\\
&\quad + BCE + 2B^2D - 2BCG + 3BGE),\\
I_{51} &= (2G - E + A)^2.
\end{aligned}$$

Let $B = 1, C = 1, D = 1$ and $G = -1$, and take the function

$$mrealroot([p_{54}, p_{53}, L_2, L_1], [E, A, F, H], 1/10^{10}, [L_5, q_{51}I_{51}]);$$

then we obtain

$$\Big[\Big[-\tfrac{9162596899}{17179869184}, -\tfrac{4581298449}{8589934592}\Big], \Big[-\tfrac{1145324613}{8589934592}, -\tfrac{286331153}{2147483648}\Big],$$

$$\Big[-\tfrac{85899345925}{17179869184}, -\tfrac{85899345915}{17179869184}\Big]\Big], \Big[\tfrac{85899345915}{17179869184}, \tfrac{85899345925}{17179869184}\Big], \quad -, -\Big].$$

This real root makes the first six focal values to be independent. Therefore, system (4.5) has six small amplitude limit cycles.

References

[1] B. Buchberger. An algorithm for finding a basis for the residue class of zero-dimensional polynomial ideal. *Aequationes Math.* 4/3, 374–383 (1970).

[2] B.W. Char, K.O. Geddes, G.H. Gonnet et al. *Maple V — Language Reference Manual.* New York: Springer (1991).

[3] C.S. Coleman. Hilbert's 16th problem: How many cycles? In: *Differential Equation Models* (M. Braun et al., eds.), pp. 279–297. Berlin: Springer (1982).

[4] G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition — Twenty years of progress. In: *Quantifier Elimination and Cylindrical Algebraic Decomposition* (B. Caviness and J.R. Johnson, eds.), pp. 8–23. Wien, New York: Springer (1998).

[5] D. Cox, J. Little, D. O'Shea. *Ideals, Varieties, and Algorithms* (2nd edn.). New York, Berlin: Springer (1996).

[6] D. Cox, J. Little, D. O'Shea. *Using Algebraic Geometry.* New York, Berlin: Springer (1996).

[7] G. Feng, S. Zhang. Reducing the multivariate polynomial system to eigentvalue problem. In: *Proc. Int. Workshop Math. Mech.* (W.-T. Wu and M. Cheng, eds.), pp. 19–27. Beijing: Int. Academic Publ. (1992).

[8] J.R. Johnson. Algorithm for polynomial real root isolation. In: *Quantifier Elimination and Cylindrical Algebraic Decomposition* (B. Caviness and J.R. Johnson, eds.), pp. 269–299. New York: Springer (1998).

[9] T.Y. Li, T. Sauer, J.A. Yorke. Numerically determining solutions of systems of polynomial equations. *Bull. Amer. Math. Soc.* 18, 173–177 (1988).

[10] N.G. Lloyd, J.M. Pearson, E. Saez et al. Limit cycles of a cubic Kolmogorov system. *Appl. Math. Lett.* 9, 15–18 (1996).

[11] Z. Lu, B. He. Multiple stable limit cycles for a cubic Kolmogorov prey-predator system. *Chinese J. Engin. Math.* 18, 115–117 (2001).

[12] Z. Lu, B. He, Y. Luo. *An Algorithm of Real Root Isolation for Polynomial Systems with Applications.* Bejing: Science Press (2004).

[13] Z. Lu, B. He, Y. Luo, L. Pan. An algorithm of real root isolation for polynomial systems. *MM Research Preprints* (Academia Sinica) 20, 187–198 (2001).

[14] Z. Lu, B. He, Y. Luo, L. Pan. The construction of small amplitude limit cycles for differential polynomial systems. Preprint (2000).

[15] Z. Lu, Y. Luo. Two limit cycles in three-dimensional Lotka-Volterra systems. *Comput. Math. Appl.* 44, 51–66 (2002).

[16] Z. Lu, Y. Luo. Three limit cycles for a three-dimensional Lotka-Volterra competitive system with a heteroclinic cycle. *Comput. Math. Appl.* 46, 281–288 (2003).

[17] J.F. Ritt. *Differential Algebra.* New York: Amer. Math. Soc. (1950).

[18] S. Smale. The fundamental theorem of algebra and complexity theory. *Bull. Amer. Math. Soc.* 4, 1–36 (1981).

[19] D. Wang. A class of cubic differential systems with 6-tuple focus. *J. Diff. Eqns.* 87, 305–315 (1990).

[20] W.-T. Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Sci. Sinica* 21, 150–172 (1978).

[21] W.-T. Wu. *Mechanical Theorem Proving in Geometries: Basic Principle.* Wien, New York: Springer (1994).

[22] W.-T. Wu. *Mathematics Mechanization.* Beijing: Science Press/Kluwer (2000).

Zhengyi Lu, Bi He and Yong Luo
Department of Mathematics
Wenzhou University
Wenzhou 325003, China
e-mail: zhengyilu@hotmail.com
sanjiaoxingh@sohu.com
yluo@mmrc.iss.ac.cn

Lu Pan
Department of Mathematics
Sichuan University
Chengdu 610064, China
e-mail: plbox@163.com

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 149–166

An Algebraic Method for Separating Close-Root Clusters and the Minimum Root Separation

Tateaki Sasaki and Fujio Kako

Abstract. Given a univariate polynomial over **C**, we discuss two issues, an algebraic method for separating a factor of mutually close roots from the polynomial, and a reasonable formula for the minimum root separation, by assuming that the close roots form well-separated clusters. The technique we use is very simple and effective; we move the origin near to the center of a close-root cluster, and then we are able to treat the other roots collectively, reducing the problem to a very simple one. Following this idea, we present a very simple and stable algebraic method for separating the close-root cluster, derive two lower-bound formulas for the distance between two close roots, and obtain a fairly simple lower bound of the minimum root separation of polynomials over **C**.

Keywords. Close root, close-root cluster, minimum root separation, separation of close roots.

1. Introduction

In this paper, assuming that a given univariate polynomial contains well-separated clusters of close roots and that the coefficients of the given polynomial can be computed to any required accuracy, (1) we present a new algebraic method for separating the close-root clusters, and (2) we investigate the minimum root separation. Although these two issues are different, we discuss them in a single article because we employ the same approach.

Our technique of attacking the above issues is very simple and effective; it was devised in [TS00] first and used in [IS04] etc. If we move the origin near to the center of a close-root cluster, the coefficients of the shifted polynomial show a peculiar

Work supported in part by the Japanese Ministry of Education, Science and Culture under Grants 15300002.

behavior. Using this behavior, we can treat the roots other than those in the cluster collectively, and the problem is reduced to a much simpler one. Furthermore, with this technique, we can usually obtain fairly accurate inequalities, where by *accurate inequality* we mean that the quantities in the left hand side (l.h.s.) and the right hand side (r.h.s.) are not much different.

Computation of the roots of a univariate polynomial is a very old issue, but it is still posing problems to researchers. In 1990's, several semi-algebraic methods were proposed for factoring a univariate polynomial into two factors numerically hence approximately. Sakurai, Sugiura and Torii [SST92] proposed a method which is based on an Hermite interpolation. Pan [Pan95, Pan96, Pan01] proposed a method using Graeffe's technique and revealed a very good computational complexity of the method. Repeating this factorization recursively, we obtain linear factors hence the roots. The algorithms seem to be quite effective for many polynomials, but actually they become unstable if the given polynomial contains close roots.

In this paper, we consider ill-conditioned polynomials which contain well-separated clusters of close roots. In order to separate factors of close roots, Sasaki and Noda [SN89] proposed an algorithm of approximate square-free decomposition, and Hribernig and Stetter [HS97] presented a similar method. These authors used the polynomial remainder sequence (PRS) in their algorithms. In this paper, assuming that the PRS has been computed by a suitable method, we present a very simple and very stable algorithm for separating a factor containing only a cluster of mutually close roots. The algorithm presented is crucially based on the above-mentioned peculiar behavior of the shifted polynomial.

In computer algebra, there are several quantities for which the theoretical lower or upper bounds differ from the actual values by many orders of magnitudes. The minimum root separation is one of such quantities. Let $A(x)$ be a given square-free polynomial over $\mathbf{C}$ or $\mathbf{Z}$, of degree $n \geq 3$, having the roots $\alpha_1, \ldots, \alpha_n$, where $\alpha_i \neq \alpha_j$ $(\forall i \neq j)$. The minimum root separation, or $\mathrm{sep}(A)$ in short, is defined to be $\mathrm{sep}(A) = \min\{|\alpha_i - \alpha_j| \mid 1 \leq i < j \leq n\}$.

So far, many formulas for the lower bound of $\mathrm{sep}(A)$ were presented; see [Mig92]. For polynomials over $\mathbf{Z}$, Collins and Horowitz [CH74] derived the bound $\mathrm{sep}(A) > \frac{1}{2}\, e^{-n/2}\, n^{-3n/2}\, \|A\|_\infty^{-n}$, where $\|A\|_p$ is the p-norm. A much better lower bound is given by Mignotte [Mig92]: $\mathrm{sep}(A) > n^{-(n+2)/2} D^{1/2}\, \|A\|_2^{-(n-1)}$, where D denotes the discriminant of $A(x)$. Unfortunately, these theoretical bounds are extraordinary smaller than experimental values. In fact, after many experiments, Collins [Col01] conjectured that $\mathrm{sep}(A) > n^{-n/4} \|A\|_\infty^{-n/2}$. In deriving the theoretical formulas, close roots are not considered so far. The minimum root separation is determined by the closest roots, hence, we are absolutely necessary to take the close roots into account.

In this paper, we will present a new approach to the minimum root separation. The approach is again based on the peculiar behavior of the shifted polynomial. It should be noted that the given polynomial must be treated as accurately as required (by high precision arithmetic in several steps of algorithm). In fact, if the coefficients are perturbed by relative magnitude ε, $0 < \varepsilon \ll 1$, then m multiple or very close roots may be moved as large as $O(\varepsilon^{1/m})$, hence the minimum root separation will also be changed largely. On the other hand, many authors took an approach to compute multiple roots pretty accurately without using high precision arithmetic, see [Zen03] for example. In this approach, it is assumed (often implicitly) that the given polynomial has multiple roots and no close roots; in other words, a cluster of close roots of a polynomial with truncated coefficients are regarded as multiple roots. Without such an assumption, we cannot compute "multiple" roots by the fixed-precision arithmetic; in fact, the concept of multiple roots can never hold for polynomials with inexact coefficients (we should replace it by "approximately multiple" roots).

In Sect. 2, we review two theorems for distinguishing close roots distributed near the origin from the other roots. We append two new theorems for the cluster of two close roots. The reader will see our technique from the proofs of these theorems. In Sect. 3, we define a normalized PRS and explain characteristic behaviors of the sequence when the given polynomial has close roots. Then, we review a method for finding the locations of the clusters. Finally, we present a very simple and very stable method for separating a factor containing only the close roots in a cluster. In Sect. 4, we derive two lower bounds for the distance between two close roots and obtain a fairly simple lower bound for the minimum root separation. In Sect. 5, we point out open problems being concerned with this work.

2. Gap Theorems on the Roots

In this section, we review two theorems on a cluster of close roots of $A(x)$:

$$A(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = a_n(x-\alpha_1)\cdots(x-\alpha_n). \tag{2.1}$$

Furthermore, we append two new theorems for special cases. First, we show a well-known lemma; see [Mig92] for the proof.

Lemma 1. *Let a_n and a_0 be not zero and the roots be ordered as $|\alpha_1| \le \cdots \le |\alpha_n|$. Then, $|\alpha_1|$ and $|\alpha_n|$ are bounded as follows:*

$$\frac{|a_0|}{|a_0| + \max\{|a_1|,\ldots,|a_n|\}} \le |\alpha_1| \le |\alpha_n| \le \frac{|a_n| + \max\{|a_{n-1}|,\ldots,|a_1|\}}{|a_n|}. \tag{2.2}$$

By using this lemma, the following theorems were proved; see [TS00] or [ST02] for the proof of Theorem 1 and [IS04] for the proof of Theorem 2.

Theorem 1 (Sasaki and Terui). *Let $\bar{A}(x) \in \mathbf{C}[x]$ be*

$$\bar{A}(x) = \bar{a}_n x^n + \cdots + \bar{a}_{m+1}x^{m+1} + 1\cdot x^m + \bar{e}_{m-1}x^{m-1} + \cdots + \bar{e}_0, \tag{2.3}$$

where the coefficients satisfy

$$\begin{cases} \max\{|\bar{a}_n|,\ldots,|\bar{a}_{m+1}|\} = 1, \\ \bar{e} \overset{\text{def}}{=} \max\{|\bar{e}_{m-1}|^{1/1}, |\bar{e}_{m-2}|^{1/2},\ldots,|\bar{e}_0|^{1/m}\} \ll 1. \end{cases} \tag{2.4}$$

If $\bar{e} < 1/9$ then $\bar{A}(x)$ has m small roots inside a disc $\bar{D}_{\mathrm{in}}$ of radius $\bar{R}_{\mathrm{in}}$ and other $n-m$ roots outside a disc $\bar{D}_{\mathrm{out}}$ of radius $\bar{R}_{\mathrm{out}}$, located at the origin, where

$$\bar{R}_{\mathrm{in(out)}} = \frac{(1+3\bar{e}) - (+)\sqrt{(1+3\bar{e})^2 - 16\bar{e}}}{4}. \tag{2.5}$$

Corollary 1. *Each root in the close-root cluster located at the origin, of $\bar{A}(x)$ is separated from other $n-m$ roots at least by $\bar{R}_{\mathrm{out}} - \bar{R}_{\mathrm{in}} = \frac{1}{2}\sqrt{(1+3\bar{e})^2 - 16\bar{e}}$.*

Theorem 2 (Inaba and Sasaki). *Let $m = 1$ in (2.3) and put $\hat{e} = |\bar{e}_0|$. If $\hat{e} < 1/(3+2\sqrt{2})$ then $\bar{A}(x)$ has one small root inside a disc $\hat{D}_{\mathrm{in}}$ of radius $\hat{R}_{\mathrm{in}}$ and other $n-1$ roots outside a disc $\hat{D}_{\mathrm{out}}$ of radius $\hat{R}_{\mathrm{out}}$, located at the origin, where*

$$\hat{R}_{\mathrm{in(out)}} = \frac{(1+\hat{e}) - (+)\sqrt{(1+\hat{e})^2 - 8\hat{e}}}{4}. \tag{2.6}$$

Corollary 2. *For $m = 1$, the smallest root around the origin of $\bar{A}(x)$ is separated from other $n-1$ roots at least by $\hat{R}_{\mathrm{out}} - \hat{R}_{\mathrm{in}} = \frac{1}{2}\sqrt{(1+\hat{e})^2 - 8\hat{e}}$.*

$\bar{R}_{\mathrm{in}}$ and $\bar{R}_{\mathrm{out}}$ (and $\hat{R}_{\mathrm{in}}$ and $\hat{R}_{\mathrm{out}}$, too) are two roots of a quadratic polynomial. $\hat{R}_{\mathrm{in}}$ and $\hat{R}_{\mathrm{out}}$ were obtained first by Wang and Han in [WH90], in a study of Newton's method for computing a root of univariate polynomial. $\bar{R}_{\mathrm{in}}$ was obtained also by Yakoubsohn [Yak00] by a different approach, but he did not obtain $\bar{R}_{\mathrm{out}}$. How accurately the formula (2.5) bounds the actual roots was investigated numerically by Sasaki and Terui [ST02], which revealed that the formula bounds the actual roots fairly well.

2.1. Special two Cases of $m = 2$

In this paper, we are interested in the case that only two close roots form a cluster. In this subsection, we specialize the above theorems for the case of $m = 2$.

We first consider the case that the origin is very close to one of the two close roots. If we perform a scale transformation so that the cluster size becomes $O(1)$, the polynomial $A(x)$ will be transformed to the following regularized form:

$$\left\{ \begin{array}{c} \bar{A}_2(x) = \bar{a}_n d^{n-2} x^n + \cdots + \bar{a}_3 d x^3 + \bar{a}_2 x^2 + x + \bar{e}_0, \\ 0 < d \ll 1, \quad 0 < |\bar{e}_0| \ll 1, \\ \max\{|\bar{a}_n|, \ldots, |\bar{a}_3|\} = |\bar{a}_2| = 1. \end{array} \right. \tag{2.7}$$

We can transform $\bar{A}(x)$ in (2.3), with $m = 2$, to the polynomial $\bar{A}_2(x)$ in (2.7), as follows. If $|\bar{e}_1|^2 \gg |\bar{e}_0|$ then put $\bar{A}_2(x) = \bar{A}(\bar{e}_1 x)/\bar{e}_1^2$. Otherwise, compute $\bar{A}'(x) = \bar{A}(x+\alpha') \overset{\mathrm{def}}{=} a'_n x^n + \cdots + a'_2 x^2 + a'_1 x + a'_0$, with $\alpha' = (-\bar{e}_1 + \sqrt{\bar{e}_1^2 - 4\bar{e}_0})/2$, and put $\bar{A}_2(x) = \bar{A}'(a'_1 x/a'_2) \cdot a'_2/(a'_1)^2$. Note that $|\alpha'| \ll 1$ because $\max\{|\bar{e}_1|, |\bar{e}_0|^{1/2}\} \ll 1$.

$\bar{A}_2(x)$ has one small root near the origin, let it be $\hat{\gamma}$. Let $\check{\gamma}$ be a root of $\bar{A}_2(x)$, other than $\hat{\gamma}$, then we have $|\check{\gamma}| \gtrsim 0.5$. For $\bar{A}_2(x)$, we can strengthen Theorem 2 as follows.

Theorem 3. *Put $\hat{e} = |\bar{e}_0|$ for simplicity. If $\hat{e} < 1/[(2+d) + 2\sqrt{1+d}]$ then $\bar{A}_2(x)$ has one small root inside a disc $\hat{D}_{\rm in}$ of radius $\hat{R}_{\rm in}$ and other $n-1$ roots outside a disc $\hat{D}_{\rm out}$ of radius $\hat{R}_{\rm out}$, located at the origin, where*

$$\hat{R}_{\rm in(out)} = \frac{(1+d\hat{e}) - (+)\sqrt{(1+d\hat{e})^2 - 4(1+d)\hat{e}}}{2(1+d)}. \tag{2.8}$$

Corollary 3. *The smallest root $\hat{\gamma}$ around the origin of $\bar{A}_2(x)$ is separated from other $n-1$ roots at least by $\sqrt{(1+d\hat{e})^2 - 4(1+d)\hat{e}}/(1+d)$.*

Proof. Equality $\bar{A}_2(\hat{\gamma}) = 0$ gives us $\hat{\gamma}\cdot(\bar{a}_n d^{n-2}\hat{\gamma}^{n-1} + \cdots + \bar{a}_3 d\hat{\gamma}^2 + \bar{a}_2\hat{\gamma} + 1) = -\bar{e}_0$. Since $|\hat{\gamma}| \ll 1$, we obtain

$$\begin{aligned} |\hat{\gamma}| \ &= \ \hat{e} \ / \ |1 + \bar{a}_2\hat{\gamma} + \bar{a}_3 d\hat{\gamma}^2 \cdots + \bar{a}_n d^{n-2}\hat{\gamma}^{n-1}| \\ &\le \ \hat{e} \ / \ \{1 - |\hat{\gamma}| - d|\hat{\gamma}|^2 - \cdots - d^{n-2}|\hat{\gamma}|^{n-1}\} \\ &< \ \hat{e} \ / \ \{1 - |\hat{\gamma}|/(1 - d|\hat{\gamma}|)\} \\ &\Longrightarrow \ (1+d)|\hat{\gamma}|^2 - (1+d\hat{e})|\hat{\gamma}| + \hat{e} > 0. \end{aligned}$$

The last inequality holds only if $(1+d\hat{e})^2 - 4(1+d)\hat{e} > 0$, or $\hat{e} < 1/[(2+d) + 2\sqrt{1+d}]$. With this condition, the above last inequality gives us the bound $\hat{R}_{\rm in}$ for $|\hat{\gamma}|$.

Next, consider $d^2\bar{A}_2(x/d) \overset{\rm def}{=} \tilde{A}_2(x) = \bar{a}_n x^n + \cdots + \bar{a}_2 x^2 + dx + d^2\bar{e}_0$. $\tilde{A}_2(x)$ has a root $d\check{\gamma}$, and we put $\gamma = d\check{\gamma}$ for simplicity. Dividing $\tilde{A}_2(\gamma) = 0$ by γ, we obtain equality $\bar{a}_n\gamma^{n-1} + \cdots + \bar{a}_2\gamma + \bar{a}_1 = 0$, where $\bar{a}_1 = d(1 + d\bar{e}_0/\gamma)$. We see $|\bar{a}_1| \simeq d \ll 1$ because $|\check{\gamma}| \gtrsim 0.5$. We regard this equality as an equation in γ, of degree $n-1$ with the constant term $\bar{a}_1$, and apply formula (2.2) to it. (We can state this situation as follows. Consider a set of polynomials $\{\bar{a}_n z^{n-1} + \cdots + \bar{a}_2 z + \bar{a}_1 \mid |\bar{a}_1| \le \bar{a}\}$, where $\bar{a}$ is so chosen that the set contains a polynomial having the root γ. Since $\bar{a}_1$ is a number, we can apply formula (2.2) to every polynomial in the set.) Then, we obtain

$$\begin{aligned} |\gamma| \ &\ge \ \frac{1}{1 + 1\,/\,|\bar{a}_1|} \ \ge \ \frac{1}{1 + 1/(d - d^2\hat{e}/|\gamma|)} \\ &\Longrightarrow \ (1+d)|\gamma|^2 - d(1+d\hat{e})|\gamma| + d^2\hat{e} \ge 0 \\ &\text{or} \quad (1+d)|\check{\gamma}|^2 - (1+d\hat{e})|\check{\gamma}| + \hat{e} \ge 0. \end{aligned}$$

We have obtained the same polynomial for both $|\hat{\gamma}|$ and $|\check{\gamma}|$. Since $\check{\gamma}$ is not the smallest root, we obtain the bound $\hat{R}_{\rm out}$ for $|\check{\gamma}|$. The $\hat{\gamma}$ is separated from other $n-1$ roots by at least $\hat{R}_{\rm out} - \hat{R}_{\rm in}$, which proves the corollary. □

Next, we consider the case that the origin is near to the center of close-root cluster. That is, instead of $\bar{A}_2(x)$ in (2.7), we consider $\bar{A}_2(x)$ regularized as follows:

$$\left\{ \begin{array}{ll} \bar{A}_2(x) = \bar{a}_n x^n + \cdots + \bar{a}_3 x^3 + x^2 + \bar{e}_1 x + \bar{e}_0, & |\bar{e}_1| < |\bar{e}_0|, \\ \max\{|\bar{a}_n|, \ldots, |\bar{a}_3|\} = 1, \qquad \bar{e} \overset{\rm def}{=} \max\{|\bar{e}_1|, |\bar{e}_0|^{1/2}\} \ll 1. & \end{array} \right. \tag{2.9}$$

We can transform $\bar{A}(x)$ in (2.3), with $m = 2$, to the polynomial $\bar{A}_2(x)$ in (2.9), as follows. Compute $\bar{A}'(x) = \bar{A}(x - \bar{e}_1/2) \overset{\rm def}{=} a'_n x^n + \cdots + a'_2 x^2 + a'_1 x + a'_0$, and put $\bar{A}_2(x) = \bar{A}'(x/\eta)\cdot\eta^2/a'_2$, where $\eta = \max\{|a'_3/a'_2|^{1/1}, |a'_4/a'_2|^{1/2}, \ldots, |a'_n/a'_2|^{1/(n-2)}\}$.

$\bar{A}_2(x)$ has two small roots around the origin, let one of them be $\hat{\gamma}$, and any other root $\breve{\gamma}$ of $\bar{A}_2(x)$ is such that $\breve{\gamma} \gtrsim 0.5$.

Theorem 4. *Let a polynomial $P_2(r)$ be defined as*

$$P_2(r) = 2r^3 - (1 + |\bar{e}_1|)r^2 + (|\bar{e}_1| - |\bar{e}_0|)r + |\bar{e}_0|. \tag{2.10}$$

If $\bar{e}_1$ and $\bar{e}_0$ in (2.9) are such that $P_2(r)$ has two real positive roots $\bar{R}_{\text{in}}$ and $\bar{R}_{\text{out}}$, with $\bar{R}_{\text{in}} < \bar{R}_{\text{out}}$, then $\bar{A}_2(x)$ has two small roots inside a disc $\bar{D}_{\text{in}}$ of radius $\bar{R}_{\text{in}}$ and $n-2$ roots outside a disc $\bar{D}_{\text{out}}$ of radius $\bar{R}_{\text{out}}$, located at the origin. The conditions for $P_2(r)$ having two real positive roots are

$$\begin{cases} \textit{Condition } 1: & |\bar{e}_1| < |\bar{e}_0|, \\ \textit{Condition } 2: & R < 0, \end{cases} \tag{2.11}$$

where $R = |\bar{e}_1|^4 - |\bar{e}_1|^3(6 - 2|\bar{e}_0|) + |\bar{e}_1|^2(1 - 4|\bar{e}_0| + |\bar{e}_0|^2) - |\bar{e}_1|(26|\bar{e}_0| - 14|\bar{e}_0|) + (4|\bar{e}_0| - 71|\bar{e}_0|^2 + 8|\bar{e}_1|^3)$.

Corollary 4. *The two small roots around the origin, of $\bar{A}_2(x)$ are separated from other $n-2$ roots at least by*

$$\bar{R}_{\text{out}} - \bar{R}_{\text{in}}. \tag{2.12}$$

Proof. $\bar{A}_2(x)$ has a root $\breve{\gamma}$. Denoting $\breve{\gamma}$ by γ and dividing $\bar{A}_2(\gamma)$ by γ^2, we obtain $0 = \bar{a}_n\gamma^{n-2} + \cdots + \bar{a}_3\gamma + \bar{a}_2$, where $\bar{a}_2 = 1 + \bar{e}_1/\breve{\gamma} + \bar{e}_0/\breve{\gamma}^2$. We see $|\bar{a}_2| \simeq 1$ because $|\breve{\gamma}| \gtrsim 0.5$. We regard the above r.h.s. expression as a polynomial in γ of degree $n-2$ with the constant term $\bar{a}_2$, and apply formula (2.2) to it. Then, we obtain the following inequality for $\breve{\gamma}$:

$$\begin{aligned} |\breve{\gamma}| \;\geq\; & \frac{1}{1 + 1\,/\,|\bar{a}_2|} \;\geq\; \frac{1}{1 + 1\,/\,(1 - |\bar{e}_1|/|\breve{\gamma}| - |\bar{e}_0|/|\breve{\gamma}|^2)} \\ \Longrightarrow \quad & 2|\breve{\gamma}|^3 - (1 + |\bar{e}_1|)|\breve{\gamma}|^2 + (|\bar{e}_1| - |\bar{e}_0|)|\breve{\gamma}| + |\bar{e}_0| \;\geq\; 0. \end{aligned}$$

Next, consider $\bar{A}_2(ex)/e^2 \overset{\text{def}}{=} \tilde{A}_2(x) = \bar{a}_n e^{n-2}x^n + \cdots + \bar{a}_3 e x^3 + x^2 + (\bar{e}_1/e)x + (\bar{e}_0/e^2)$, where $e = \bar{e}$. $\tilde{A}_2(x)$ has a root $\hat{\gamma}/e$. Putting $\gamma = \hat{\gamma}/e$, we have $\gamma^2 \cdot \{\bar{a}_n(e\gamma)^{n-2} + \cdots + \bar{a}_3(e\gamma) + 1\} = -(\bar{e}_1/e)\gamma - (\bar{e}_0/e^2)$. Since we have $e|\gamma| = |\hat{\gamma}| \ll 1$, we obtain the following inequality for $\hat{\gamma}$:

$$\begin{aligned} (|\bar{e}_1|/e)|\gamma| + (|\bar{e}_0|/e^2) \;&\geq\; |\gamma|^2 \cdot (1 - |e\gamma| - \cdots - |e\gamma|^{n-2}) \\ &>\; |\gamma|^2 \cdot \{1 - e|\gamma|/(1 - e|\gamma|)\} \\ \Longrightarrow \quad 2e^3|\gamma|^3 - (1 + |\bar{e}_1|)e^2|\gamma|^2 + (|\bar{e}_1| - |\bar{e}_0|)e|\gamma| + |\bar{e}_0| \;&>\; 0 \\ \text{or} \quad 2|\hat{\gamma}|^3 - (1 + |\bar{e}_1|)|\hat{\gamma}|^2 + (|\bar{e}_1| - |\bar{e}_0|)|\hat{\gamma}| + |\bar{e}_0| \;&>\; 0. \end{aligned}$$

We have obtained the same polynomial $P_2(r)$ for both $\hat{\gamma}$ and $\breve{\gamma}$. Since $P_2(0) = |\bar{e}_0| > 0$, $P_2(r)$ has at least one negative root. Since $|\hat{\gamma}|$ and $|\breve{\gamma}|$ must be two roots of $P_2(r)$, we have conditions in (2.11). □

Condition 2 in (2.11) is complicated to obtain the general solution, so we estimate the values of $\bar{e}$ and $\bar{R}_{\text{in}}$ for the following three cases.

$$\begin{array}{llll} \text{Case 1}: & |\bar{e}_1| = 0 \text{ and } |\bar{e}_0| = \bar{e}^2 & \Longrightarrow & \bar{e} \approx 0.23812, \quad \bar{R}_{\rm in} \approx 0.3596. \\ \text{Case 2}: & 5|\bar{e}_1| = |\bar{e}_0| = \bar{e}^2 & \Longrightarrow & \bar{e} \approx 0.23002, \quad \bar{R}_{\rm in} \approx 0.3566. \\ \text{Case 3}: & 2|\bar{e}_1| = |\bar{e}_0| = \bar{e}^2 & \Longrightarrow & \bar{e} \approx 0.21938, \quad \bar{R}_{\rm in} \approx 0.3527. \end{array}$$

The value of $\bar{R}_{\rm in}$ in Theorem 4 is much larger that those in Theorems 1 and 2. However, Theorem 4 is less useful practically than Theorems 1 and 2, because the first condition in (2.11) is quite restrictive.

3. Separating Clusters of Close Roots

In this section, by quo(A, B), rem(A, B) and lc(A), we denote the quotient and remainder of A divided by B and the leading coefficient of A, respectively. By $\|P\|$ we denote the infinity norm of polynomial P.

First of all, we note that the theorems in Sect. 2 can be generalized directly to polynomials with inexact coefficients, so long as the error bound of each coefficient is known. Here, we state the generalization for Theorem 1 only.

Proposition 1. *Let the coefficients* $\bar{a}_n, \ldots, \bar{a}_{m+1}, \bar{e}_{m-1}, \ldots, \bar{e}_0$ *of* $\bar{A}(x)$ *in Theorem* 1 *contain small errors which are bounded respectively by* $\varepsilon_n, \ldots, \varepsilon_{m+1}, \varepsilon_{m-1}, \ldots, \varepsilon_0$. *Theorem* 1 *is valid if we regularize* $\bar{A}(x)$ *as* $\max\{|\bar{a}_n|+\varepsilon_n, \ \ldots, \ |\bar{a}_{m+1}|+\varepsilon_{m+1}\} = 1$ *and define* $\bar{e}$ *as* $\bar{e} = \max\{(|\bar{e}_{m-1}|+\varepsilon_{m-1})^{1/1}, \ (|\bar{e}_{m-2}|+\varepsilon_{m-2})^{1/2}, \ \ldots, \ (|\bar{e}_0|+\varepsilon_0)^{1/m}\}$.

We assume that $A(x)$ is monic and regularized as

$$a_n = \max\{|a_{n-1}|, \ldots, |a_0|\} = 1. \tag{3.1}$$

With this regularization, Lemma 1 tells us that any root α of $A(x)$ is bounded as $|\alpha| \le 2$. Therefore, if two different roots α_i and α_j of $A(x)$ are such that $|\alpha_i - \alpha_j| \ll 1/n$ then we can say that α_i and α_j are *mutually close roots of closeness* $|\alpha_i - \alpha_j|$. We consider the case that close roots of $A(x)$ form clusters of different sizes $O(\delta_1), \ldots, O(\delta_\tau)$, with $1/n \gg \delta_1 \gg \cdots \gg \delta_\tau$ (τ may be 1), and that each cluster of size $O(\delta_i)$ is separated from other roots by distance $\gg \delta_i$. Let the number of close roots of closeness $\le O(\delta_i)$ be m_i ($i = 1, \ldots, \tau$), and the close roots of closeness $O(\delta_i)$ be distributed among ℓ_i clusters ($i = 1, \ldots, \tau$). A bigger cluster may contain several smaller clusters. We define the *center of the cluster* to be the average value of the close roots in the cluster.

3.1. Normalized PRS (Polynomial Remainder Sequence)

Putting $P_1 = A(x)$, $P_2 = \frac{1}{n}\,\mathrm{d}A/\mathrm{d}x$, $S_1 = T_2 = 1$ and $S_2 = T_1 = 0$, we generate a PRS $(P_1, P_2, P_3, P_4, \ldots)$ and cofactor sequences $(S_1, S_2, S_3, S_4, \ldots)$ and $(T_1, T_2, T_3, T_4, \ldots)$, by the following formulas:

$$\left\{ \begin{array}{rcl} q_j & := & \mathrm{quo}(P_{j-1}, P_j), \\ P_{j+1} & := & (P_{j-1} - q_j P_j)/w_j, \\ S_{j+1} & := & (S_{j-1} - q_j S_j)/w_j, \\ T_{j+1} & := & (T_{j-1} - q_j T_j)/w_j \end{array} \right. \qquad (j = 2, 3, \ldots), \tag{3.2}$$

where w_j is a number to be chosen to satisfy

$$\max\{\mathrm{lc}(S_{j+1}), \mathrm{lc}(T_{j+1})\} = 1 \quad (j = 2, 3, \ldots). \tag{3.3}$$

We call the PRS generated by the formulas in (3.2) **normalized PRS**.

Remark 1. We call a quantity Q **shift-invariant** if $Q = Q'$, where Q' is the same quantity as Q but computed after shifting the origin arbitrarily. The l.h.s. quantity in (3.3) is shift-invariant, hence $\mathrm{lc}(P_{j+1})$ $(j = 2, 3, \ldots)$ are shift-invariant. In [SS97] and [Sas03], the normalization formula $\max\{\|S_{j+1}\|, \|T_{j+1}\|\} = 1$ is used. If $A(x)$ is regularized, the PRS's normalized by both formulas are almost the same.

Let indices $k_1, k_2, \ldots, k_\tau$ be such that

$$\deg(P_{k_i}) = m_i - \ell_i \quad (i = 1, 2, \ldots, \tau). \tag{3.4}$$

$\deg(P_{k_i})$ is equal to the number of roots of $\mathrm{d}A/\mathrm{d}x$, of closeness $\le O(\delta_i)$, distributed among ℓ_i clusters, and P_{k_i} is an approximate common divisor of $A(x)$ and $\mathrm{d}A/\mathrm{d}x$.

The normalized PRS behaves as follows, see [SS89] and [Sas03].

(1) We have $\|P_j\| = O(\delta_1^0)$ for $j \le k_1$ (i.e., $\|P_j\|$ does not decrease much until the remainder becomes an approximate GCD of A and $\mathrm{d}A/\mathrm{d}x$).
(2) For each $i = 1, \ldots, \tau$, we have $\|P_{k_i+1}\|/\|P_{k_i}\| = O(\delta_i^2)$ (i.e., P_{k_i} is an approximate common divisor of A and $\mathrm{d}A/\mathrm{d}x$, of tolerance $O(\delta_i^2)$).
(3) If $A(x)$ contains only one cluster of size $O(\delta_1)$, we call the PRS **single-cluster type**. For $j > k_1$, the PRS of single-cluster type behaves as $\|P_{k_1+j}\| = O(\delta_1^{2j})$ for $j = 1, 2, \ldots$ (until we encounter smaller clusters).
(4) If $A(x)$ contains two or more clusters of size $O(\delta_1)$, we call the PRS **multiple-cluster type**. For $j > k_1$, the PRS of multiple-cluster type behaves as $\|P_{k_1+j+1}\|/\|P_{k_1+1}\| = O(\delta_1^0)$ for $1 \le j < \ell_1$, $\|P_{k_1+\ell_1+1}\|/\|P_{k_1+1}\| = O(\delta_1^2)$, and so on. That is, ℓ successive remainder computations strip ℓ close roots from the ℓ clusters, one root from each cluster, without changing the norm of remainders much until the ℓ close roots are stripped. The PRS may become of single-cluster type after being stripped off all the close roots of closeness $O(\delta_1)$.

It should be emphasized that single- and multiple-cluster types are clearly distinguished from each other by the behavior of normalized PRS.

Remark 2. From the viewpoint of elimination, the elements of PRS are unique up to constant factors. From the viewpoint of computation, however, the accuracy of the result depends very much on the algorithm used; in fact, the conventional Euclidean algorithm causes large cancellations errors if small leading coefficients appear during the computation. This instability can be removed largely (but not always) by performing the elimination carefully. For example, Ohsako et al. [OST97] utilized the Givens transformation which works nicely for removing the cancellation errors. One may compute an approximate GCD of $P_1(x)$ and $P_2(x)$ by a stable method such as proposed by Corless et al. [CWZ02].

3.2. Finding the Location of a Cluster

Finding the location of clusters of close roots has already been discussed in [SN89] and [HS97]. Here, we briefly survey the method.

We first consider the single-cluster case. Assume that the normalized PRS for $A(x)$ is of single-cluster type (hence, $\ell_1 = 1$), and put $\delta_1 = \delta$, $k_1 = k$ and $m_1 = m$. Let the mutually close roots in the cluster be $\alpha_1, \dots, \alpha_m$ and $\alpha_{\rm c}$ the cluster center: $\alpha_{\rm c} = (\alpha_1 + \cdots + \alpha_m)/m$. We express $P_k(x)$ and define $\alpha'_{\rm c}$ as

$$\begin{cases} P_k(x) = p_{m-1}x^{m-1} + p_{m-2}x^{m-2} + \cdots + p_0, \\ \qquad \alpha'_{\rm c} \overset{\rm def}{=} \frac{-1}{m-1}\, p_{m-2}/p_{m-1}. \end{cases} \tag{3.5}$$

We will show in Sect. 3.2 that $|\alpha'_{\rm c} - \alpha_{\rm c}| = O(\delta^2)$, hence $\alpha'_{\rm c}$ is very close to the cluster center $\alpha_{\rm c}$. Furthermore, we can know the cluster size δ approximately by the ratio $\|P_{k+1}\|/\|P_k\| = O(\delta^2)$.

The fact $\|P_{k+1}\|/\|P_k\| = O(\delta^2)$ shows that the above mentioned method will work quite well for clusters of small sizes such as $\delta \lesssim 10^{-3}$ or 10^{-4}. So, in the example below, we check the method by a cluster of very large size $\sim 10^{-1}$.

Example 1. (Single-cluster type). Let $A(x)$ be as follows:

$$A = (x^2 - 1)\,(x - 0.30)\,(x - 0.31)\,(x - 0.35)\,(x^2 - 0.60x + 0.0925).$$

$A(x)$ has five close roots of closeness ~ 0.05 at $x \sim 0.3$ ($\alpha_{\rm c} = 0.312$), and the normalized PRS shows that $\|P_5\|/\|P_4\| \sim 0.0021 \approx (0.05)^2$. Determining $\alpha'_{\rm c}$ by P_4 as described above, we obtain $\alpha'_{\rm c} = 0.31139\cdots$. Shifting the origin to $\alpha'_{\rm c}$, we obtain $A'(x) \overset{\rm def}{=} A(x+\alpha'_{\rm c})$ and $P'_4 \overset{\rm def}{=} P_4(x+\alpha'_{\rm c})$ as follows:

$$\begin{aligned} A'(x) &= x^7 + 0.61977\cdots x^6 - 0.90334\cdots x^5 + 0.00300\cdots x^4 - 0.00010\cdots x^3 \\ &\quad + 0.70895\cdots \times 10^{-4}x^2 + 0.11466 \times 10^{-5}x - 0.14589\cdots \times 10^{-8}, \\ P'_4 &= -0.15544\cdots x^4 - 1.0000\cdots \times 10^{-4}x^2 + \cdots + 4.0545\cdots \times 10^{-8}. \end{aligned}$$

Observe that the x^4-term of $A'(x)$ is abnormally small ($\sim (0.05)^2$) and that x^i-terms ($i \le 3$) of $A'(x)$ decrease steadily as i decreases.

We next consider the case of multiple clusters. Assume that $A(x)$ contains ℓ clusters of close roots of closeness $O(\delta)$, with $\ell > 1$, then $A(x)$ must be of the following form:

$$A(x) = \tilde{A}(x)\,(x - \alpha'_1)^{\mu_1} \cdots (x - \alpha'_\ell)^{\mu_\ell} C(x) + O(\delta^2), \qquad \mu_1 \ge \cdots \ge \mu_\ell. \tag{3.6}$$

Here, α'_i is an approximate center of the ith cluster ($i=1,\dots,\ell$), $\tilde{A}(x)$ represents the factor of all the non-close roots, and $C(x)$ represents the factor of close roots of smaller closenesses.

Given $A(x)$, we generate the following normalized PRS for $i=1 \Rightarrow 2 \Rightarrow \cdots \Rightarrow \mu_1$ successively:

$$\begin{cases} (\,P_1^{(i)} = P^{(i-1)},\ P_2^{(i)} = \frac{1}{\deg(P^{(i-1)})}{\rm d}P^{(i-1)}/{\rm d}x,\ \dots,\ P_{k_i}^{(i)} \overset{\rm def}{=} P^{(i)},\ P_{\rm last}^{(i)}\,), \\ \qquad \|P_j^{(i)}\| = O(\delta^0)\ \ (j \le k_i), \qquad \|P_{\rm last}^{(i)}\| \le O(\delta^2), \end{cases} \tag{3.7}$$

where $P^{(0)} = A(x)$. Note that $P^{(1)}, P^{(2)}, \ldots$ are determined by large decreases of $\|P^{(1)}_{\mathrm{last}}\|, \|P^{(2)}_{\mathrm{last}}\|, \ldots$. Then, for index $i < \mu_\ell$, we have $P^{(i)} \propto (x-\alpha'_1)^{\mu_1-i}\cdots(x-\alpha'_\ell)^{\mu_\ell-i}C(x) + O(\delta^2)$. For index $\mu_\ell \le i \le \mu_1$, we have $P^{(i)} \propto (x-\alpha'_1)^{\mu_1-i}\cdots(x-\alpha'_r)^{\mu_r-i}C(x)+O(\delta^2)$, where $1 < r < \ell$. Therefore, we can obtain square-free factors of the product $(x-\alpha'_1)^{\mu_1}\cdots(x-\alpha'_\ell)^{\mu_\ell}$ by computing the quotients of $P^{(i-1)}$ by $P^{(i)}$ $(1 \le i \le \mu_1)$ successively. For example, if $\mu_1 = \cdots = \mu_r > \mu_{r+1} = \cdots = \mu_\ell$ we will obtain the following polynomials:

$$\begin{aligned}
P^{(\mu_\ell-1)} &\simeq C(x)\cdot[(x-\alpha'_1)\cdots(x-\alpha'_r)]^{\mu_1-\mu_\ell+1}\cdot[(x-\alpha'_{r+1})\cdots(x-\alpha'_\ell)],\\
P^{(\mu_1-1)} &\simeq C(x)\cdot[(x-\alpha'_1)\cdots(x-\alpha'_r)],\\
P^{(\mu_1)} &\simeq C(x).
\end{aligned}$$

Thus, we have $\mathrm{quo}(P^{(\mu_1-1)}, P^{(\mu_1)}) \approx (x-\alpha'_1)\cdots(x-\alpha'_r)$ and $\mathrm{quo}(P^{(\mu_\ell-1)}, P^{(\mu_\ell)}) \approx (x-\alpha'_1)\cdots(x-\alpha'_\ell)$. Finally, computing the roots of approximately square-free factors $(x-\alpha'_1)\cdots(x-\alpha'_r)$ and $(x-\alpha'_{r+1})\cdots(x-\alpha'_\ell)$, we can find the approximate locations of the clusters.

Remark 3. Separating the approximate factor $(x-\alpha'_1)^{\mu_1}\cdots(x-\alpha'_\ell)^{\mu_\ell}$ in (3.6) to factors of the same "multiplicity" is the most unstable step of our method; this point was discussed to some extent in [SN89]. If close-root clusters are closer, the separation becomes more unstable. This fact forces us to assume that the close-root clusters are well separated each other. This instability will be reduced much if we adopt such "least square algorithms" as were used in [Zen03].

3.3. A Basic Problem on PRS

Let $B(x)$ be a polynomial regularized as $A(x)$ in (3.1). If $A(x)$ and $B(x)$ have mutually close roots of closeness $O(\delta)$, the normalized PRS of $P_1 = A(x)$ and $P_2 = B(x)$ gives us $\|P_{k_1+1}\| = O(\delta)$ in general. In the case of $B(x) = \frac{1}{\deg(A)}\,\mathrm{d}A/\mathrm{d}x$, however, we have $\|P_{k_1+1}\| = O(\delta^2)$. This means that the normalized PRS gives us fairly accurate information on the close-root clusters. However, we have currently no answer to the following basic problem.

Problem By using the PRS, determine fairly accurate upper bounds of the size and the location of the close-root cluster of $A(x)$.

Determining fairly accurate *a priori* upper bounds of δ and α_{c} is not easy. We can, however, determine an *a posteriori* upper bound of δ so long as δ is small. We shift the origin to α'_{c}, an approximate center of the close-root cluster, determined in Sect. 3.2. We express $A(x+\alpha'_{\mathrm{c}})$ as follows:

$$A(x+\alpha'_{\mathrm{c}}) \overset{\mathrm{def}}{=} A'(x) = a'_n x^n + \cdots + a'_m x^m + \cdots + a'_0. \tag{3.8}$$

Then, we have (see [Sas03] for the proof; see also Example 1)

$$|a'_{m-1}/a'_m| = O(\delta^2), \quad |a'_{m-j}/a'_m| = O(\delta^j) \ \ (j = 2, 3, \ldots). \tag{3.9}$$

Note that the coefficients of x^{m-2}-, x^{m-3}-, …, x^0-terms of $A'(x)$ decrease steadily, hence we will be able to apply Theorem 1 to $A'(x)$.

Proposition 2. *Let* $1/d = \max\{|a'_{m+1}/a'_m|,\ |a'_{m+2}/a'_m|^{1/2},\ |a'_n/a'_m|^{1/(n-m)}\}$*, and put* $\bar{e} = e/d$*. If* $\bar{e} < 1/9$ *then the cluster size* δ *is bounded as* $\delta < \bar{R}_{\rm in}d$*, where* $\bar{R}_{\rm in}$ *is defined in* (2.5)*.*

As for $\alpha'_{\rm c}$, the problem is much more difficult. So, we give only an order estimation, which is simple. Since the leading coefficients of elements of the normalized PRS are shift-invariant, we can consider the PRS by shifting the origin to a favorite point. Thus, moving the origin to the cluster center $\alpha_{\rm c}$, we express $A(x)$ and P_k as

$$\begin{cases} A(x) \ = \ a''_n(x-\alpha_{\rm c})^n + \cdots + a''_m(x-\alpha_{\rm c})^m + \cdots + a''_0, \\ \ \ P_k \ \ = \ p''_{m-1}(x-\alpha_{\rm c})^{m-1} + p''_{m-2}(x-\alpha_{\rm c})^{m-2} + + \cdots + p''_0. \end{cases} \tag{3.10}$$

Proposition 3. *We have* $|\alpha'_{\rm c} - \alpha_{\rm c}| = O(\delta^2)$*.*

Proof. We note that $\alpha'_{\rm c} - \alpha_{\rm c} = \frac{-1}{m-1}\, p''_{m-2}/p''_{m-1}$. Since $\alpha_{\rm c}$ is the center of roots $\alpha_1, \dots, \alpha_m$, we have $a''_m = O(\delta^0)$ and $a''_{m-1} = O(\delta^2)$. As shown in [Sas03] by using the subresultant, this fact gives us $p''_{m-1} = O(\delta^0)$ and $p''_{m-2} = O(\delta^2)$, proving the proposition. □

3.4. Separating the Factor of a Close-Root Cluster

In this subsection, we consider to separate a factor of $A(x)$ to arbitrary accuracy, where the factor contains only m mutually close roots of closeness $\le O(\delta)$, with $\delta = \delta_1$. (Applying the separation algorithm recursively, we can separate any close-root cluster of size $O(\delta_i)$).

We already know $\alpha'_{\rm c}$, an approximate cluster center. First, we move the origin to $\alpha'_{\rm c}$, and compute $A'(x) = A(x+\alpha'_{\rm c})$ given in (3.8). Next, we compute the following number

$$e \ = \ \max\{|a'_{m-1}/a'_m|,\ |a'_{m-2}/a'_m|^{1/2},\ \dots,\ |a'_0/a'_m|^{1/m}\}. \tag{3.11}$$

Using e, we transform $A'(x)$ to the following regularized form:

$$\begin{cases} \bar{A}(x) \ \stackrel{\rm def}{=} \ A'(ex)/a'_m e^m \ = \ \bar{a}_n x^n + \cdots + 1\cdot x^m + \bar{a}_{m-1}x^{m-1} + \cdots + \bar{a}_0, \\ \ \ d \ \stackrel{\rm def}{=} \ \max\{|\bar{a}_n|^{1/(n-m)}, \dots, |\bar{a}_{m+1}|^{1/1}\}, \qquad \max\{|\bar{a}_{m-1}|, \dots, |\bar{a}_0|\} = 1. \end{cases} \tag{3.12}$$

By assumptions, e must be a small number of magnitude $O(\delta)$, and so is d.

We want to factor $\bar{A}(x)$ as $\bar{A}(x) = H(x)\,C(x)$, where $C(x) = x^m + c_{m-1}x^{m-1} + \cdots + c_0$. $H(x)$ can be expressed as $H(x) = 1 + d\,h_1 x + d^2 h_2 x^2 + \cdots + d^{n-m}h_{n-m}x^{n-m}$, where d is given in (3.12) and $\max\{|h_1|, |h_2|, \dots, |h_{n-m}|\} \approx 1$.

Since we assumed d to be small enough, the lower degree terms $x^m + \bar{a}_{m-1}x^{m-1} + \cdots + \bar{a}_0$ of $\bar{A}(x)$ is approximately equal to $C(x)$. Therefore, we can determine $H(x)$ and $C(x)$ iteratively as follows. As initial approximations, put $H(x) \approx H^{(0)} = 1$ and $C(x) \approx C^{(0)} = x^m + \bar{a}_{m-1}x^{m-1} + \cdots + \bar{a}_0$. Expressing $\bar{A}(x)$ as $\bar{A}(x) = [1 + \varDelta_H]\cdot[C^{(0)} + \varDelta_C]$, where $\varDelta_H = O(d)$, $\varDelta_C = O(d)$, $\deg(\varDelta_H) < n-m$ and $\deg(\varDelta_C) < m$, we have $\bar{A}(x) - C^{(0)} = \varDelta_H C^{(0)} + \varDelta_C + O(d^2)$. Thus, $\varDelta_H$ and $\varDelta_C$ up to $O(d)$ are equal to the quotient and the remainder, respectively, of $\bar{A}(x) - C^{(0)}$ divided by $C^{(0)}$: $\varDelta_H = {\rm quo}(\bar{A} - C^{(0)}, C^{(0)}) + O(d^2)$, $\varDelta_C = {\rm rem}(\bar{A} - C^{(0)}, C^{(0)}) + O(d^2)$.

Assume that we have determined $H(x)$ and $C(x)$ to order d^k as $H(x) = H^{(k)} + O(d^{k+1})$ and $C(x) = C^{(k)} + O(d^{k+1})$. We shall determine $H^{(k+1)}$ and $C^{(k+1)}$ so as to satisfy $\bar{A}(x) = H^{(k+1)}C^{(k+1)} + O(d^{k+2})$. Putting $H^{(k+1)} = H^{(k)} + \varDelta_H$ and $C^{(k+1)} = C^{(k)} + \varDelta_C$, with $\deg(\varDelta_H) < n-m$ and $\deg(\varDelta_C) < m$, we have $\bar{A} - H^{(k)}C^{(k)} = \varDelta_H\, C^{(0)} + \varDelta_C + O(d^{k+2})$ (note that $\deg(\bar{A} - H^{(k)}C^{(k)}) \le n-1$). Hence, we can determine $H^{(k+1)}$ and $C^{(k+1)}$ by the following formulas:

$$\left\{ \begin{array}{rcl} H^{(k+1)} &=& H^{(k)} + \mathrm{quo}(\bar{A} - H^{(k)}C^{(k)},\, C^{(0)}), \\ C^{(k+1)} &=& C^{(k)} + \mathrm{rem}(\bar{A} - H^{(k)}C^{(k)},\, C^{(0)}). \end{array} \right. \tag{3.13}$$

Remark 4. It should be emphasized that, since $\|C^{(0)}\| = 1 = \mathrm{lc}(C^{(0)})$, the algorithm is quite stable; if d is smaller then the algorithm is more stable and it converges faster. Note that, in the step of computing $H^{(k+1)}$ and $C^{(k+1)}$, we have only to compute $A - H^{(k)}C^{(k)}$ to order d^{k+1}: we may handle only x^{m+k}-, x^{m+k-1}-, …, x^0-terms of $\bar{A} - H^{(k)}C^{(k)}$. Hence, the above algorithm is pretty efficient. The above algorithm is of linear convergence. If we determine $\varDelta_H$ and $\varDelta_C$ to satisfy $\bar{A} - H^{(k)}C^{(k)} = \varDelta_H\, C^{(k)} + \varDelta_C\, H^{(k)} + O(d^{2k+1})$, then we get an algorithm of quadratic convergence.

Remark 5. If a coefficient of a polynomial containing m multiple roots is perturbed by a small relative magnitude ε, then the m multiple roots are split into m close roots which are distributed within a circle of radius $\le O(\varepsilon^{1/m})$. Therefore, in order to compute the roots of $C(x)$ to accuracy 2^{-p}, with p a positive integer, we must compute $C(x)$ to accuracy 2^{-mp} at least (if a close-root cluster of $C(x)$ contains smaller close-root clusters, we must compute $C(x)$ much more accurately). This means that we must compute $A'(x) = A(x+\alpha'_{\mathrm{c}})$ to accuracy 2^{-mp} at least, which is quite time-consuming. However, since the x^k-term of $\bar{A}(x)$, $k > m$, is of magnitude $O(d^{n-m-k})$, we have only to compute its coefficient to accuracy $O(2^{-mp-g}/d^{n-m-k})$, with g the number of guard bits.

Example 2. We consider the polynomial in Example 1.

We already know that $\alpha'_{\mathrm{c}} = 0.31139\cdots$. Shifting the origin to α'_{c}, we obtain the polynomial $A'(x)$ given in Example 1. From the coefficients $a'_5, a'_4, \dots, a'_0$, we obtain $e = 0.042814\cdots$, which gives us $\bar{A}(x)$ as follows:

$$\left\{ \begin{array}{rcl} \bar{A}(x) &=& 1 \cdot C^{(0)} + (-0.0020291\cdots x^2 - 0.029374x) \cdot x^5, \\ C^{(0)} &=& x^5 - 0.093589\cdots x^4 + \cdots + x^2 + \cdots - 0.011226\cdots. \end{array} \right.$$

Performing the above factor-separation algorithm, we find that the norm of difference $\varDelta^{(k)} \overset{\mathrm{def}}{=} \bar{A} - H^{(k)}C^{(k)}$ decreases as follows:

$$\begin{array}{lll} \|\varDelta^{(0)}\| \approx 2.94 \times 10^{-2} & \Rightarrow\ \|\varDelta^{(1)}\| \approx 8.00 \times 10^{-4} & \Rightarrow\ \|\varDelta^{(2)}\| \approx 4.04 \times 10^{-5} \\ \quad\cdot\ \cdot\ \cdot & \Rightarrow\ \|\varDelta^{(8)}\| \approx 4.00 \times 10^{-13} & \Rightarrow\ \|\varDelta^{(9)}\| \approx 1.60 \times 10^{-14}. \end{array}$$

We note that, since the cluster size is pretty large ($\delta \sim 10^{-1}$) hence the separation of the cluster is not easy for many separation algorithms, our algorithm works quite well and stably.

Hribernig and Stetter [HS97] constructed an algorithm for separating a close-root cluster, and the algorithm has some similarity to ours. Their algorithm is, however, complicated compared with our algorithm, because they did not utilize the fact that the coefficients $|a'_{m-1}|, |a'_{m-2}|, \ldots, |a'_0|$ of $A'(x+\alpha'_{\rm c})$ decrease steadily as expressed in (3.9).

4. On the Minimum Root Separation in a Cluster

Let $C(x)$ be a factor of $\bar{A}(x)$, corresponding to the smallest close root cluster of size $O(\delta_\tau)$, with $\deg(C) = m \geq 3$. The factor $C(x)$ is obtained by a repetition of cluster separations and scale transformations, and we can obtain $\mathrm{sep}(A)$ if $\mathrm{sep}(C)$ is determined. After regularizing $C(x)$ as in Sect. 3.4, $C(x)$ will contain no close-root cluster which can be separated by Theorem 1. In order to derive formulas on $\mathrm{sep}(C)$, however, we consider the case that $C(x)$ has only two close roots. We also assume that we have computed $\alpha'_{\rm c}$, an approximate cluster center, by the normalized PRS of $C(x)$ and $\mathrm{d}C/\mathrm{d}x$, and the origin has been shifted to $\alpha'_{\rm c}$. We denote the roots of $C(x)$ by $\gamma_1, \ldots, \gamma_m$, among which γ_1 and γ_2 are the close roots around the origin.

4.1. A Lower Bound for $|\gamma_1 - \gamma_2|$

Under some conditions, we can bound $|\gamma_1 - \gamma_2|$ by applying Theorem 1 to $C(x)$ (we may use Theorem 4 but the statement becomes complicated). We regularize $C(x)$ and define e as follows:

$$\begin{cases} \quad C(x) \ = \ c_m x^m + \cdots + c_3 x^3 + x^2 + c_1 x + c_0, \\ \max\{|c_m|, \ldots, |c_3|\} = 1, \qquad e \ \overset{\rm def}{=} \ \max\{|c_1|, |c_0|^{1/2}\}. \end{cases} \tag{4.1}$$

Theorem 5. *Let $\bar{R}_{\rm in}$ be the same as that in Theorem 1, with $\bar{e}$ replaced by e. If $e < 1/9$ and $|c_1^2 - 4c_0|/4|c_0| > \bar{R}_{\rm in}/(1 - \bar{R}_{\rm in})$ then the following inequality holds:*

$$|\gamma_1 - \gamma_2| \ > \ \frac{\sqrt{|c_1^2 - 4c_0| - 4|c_0|\,\bar{R}_{\rm in}/(1-\bar{R}_{\rm in})} \times (1 - 2\bar{R}_{\rm in})\,(1 - \bar{R}_{\rm in})}{1 + (|c_1|/2)\,(1 - \bar{R}_{\rm in})^2/(1 - 2\bar{R}_{\rm in})^3}. \tag{4.2}$$

Proof. Put $C(x) = L(x)\,x^2 + c_1 x + c_0$, with $L(x) = c_m x^{m-2} + \cdots + c_3 x + 1$, and let $\gamma \in \{\gamma_1, \gamma_2\}$. We regard γ as a root of $L(\gamma)\,x^2 + c_1 x + c_0$, then we obtain

$$(\gamma_1 - \gamma_2) - \frac{c_1}{2} \cdot \frac{L(\gamma_1) - L(\gamma_2)}{L(\gamma_1)L(\gamma_2)} \ = \ \frac{\sqrt{c_1^2 - 4c_0 L(\gamma_1)}}{2L(\gamma_1)} + \frac{\sqrt{c_1^2 - 4c_0 L(\gamma_2)}}{2L(\gamma_2)}.$$

Since $L(\gamma) = 1 + c_3\gamma + \cdots + c_m\gamma^{m-2}$ and $|\gamma| < \bar{R}_{\rm in} < 1/3$, we can bound $|L(\gamma)|$ as

$$1 - \bar{R}_{\rm in}/(1-\bar{R}_{\rm in}) \ < \ |L(\gamma)| \ < \ 1 + \bar{R}_{\rm in}/(1-\bar{R}_{\rm in}). \tag{4.3}$$

We bound $|L(\gamma_1) - L(\gamma_2)|$ as $|L(\gamma_1) - L(\gamma_2)| = |\gamma_1 - \gamma_2| \cdot |c_3 + c_4(\gamma_1 + \gamma_2) + c_5(\gamma_1^2 + \gamma_1\gamma_2 + \gamma_2^2) + \cdots| < |\gamma_1 - \gamma_2| \cdot (1 + 2|\gamma| + 4|\gamma|^2 + \cdots) = |\gamma_1 - \gamma_2|/(1 - 2|\gamma|) < |\gamma_1 - \gamma_2|/(1 - 2\bar{R}_{\rm in})$.

Putting $R(\gamma) = \sqrt{c_1^2 - 4c_0 L(\gamma)}\,/2L(\gamma)$, we search for a lower bound of $|R(\gamma_1) + R(\gamma_2)|$. Although the roots γ_1 and γ_2 are mutually related in that they are two different roots of $C(x)$, we neglect this fact and change $L(\gamma_1)$ and $L(\gamma_2)$ arbitrarily under the restriction (4.3). Then, the numerator of $|R(\gamma)|$ is bounded as

$$\sqrt{|c_1^2-4c_0L(\gamma)|} \;>\; \sqrt{|c_1^2-4c_0| - 4|c_0|\,\bar{R}_{\rm in}/(1-\bar{R}_{\rm in})}\,.$$

Noting that $L(\gamma)$ is a complex number, we bound $|1/L(\gamma_1) + 1/L(\gamma_2)|$ as

$$\left|\frac{1}{L(\gamma_1)} + \frac{1}{L(\gamma_2)}\right| \;=\; \frac{|L(\gamma_1)+L(\gamma_2)|}{|L(\gamma_1)L(\gamma_2)|} \;>\; 2\,(1-2\bar{R}_{\rm in})\,(1-\bar{R}_{\rm in}).$$

Summarizing the above bounds, we obtain the theorem. □

4.2. Basic Lemmas

The formula in Theorem 5 is rather complicated, so we search for another formula. In this and the next subsections, we regularize $C(x)$ as $A(x)$ in (3.1) and define d as follows:

$$\begin{cases} C(x) \;=\; x^m + c_{m-1}x^{m-1} + \cdots + c_2x^2 + c_1x + c_0, \\ \qquad\quad \max\{|c_{m-1}|, \ldots, |c_2|\} = 1 \gg |c_1|, |c_0|, \\ 1/d \;=\; \max\{|c_3/c_2|,\ |c_4/c_2|^{1/2},\ \ldots,\ |1/c_2|^{1/(m-2)}\}. \end{cases} \tag{4.4}$$

Put $\sigma = (\gamma_1+\gamma_2)/2$ and $\hat{\gamma} = (\gamma_1-\gamma_2)/2$, and define $H(x)$ and η as follows:

$$\begin{cases} C(x) \;=\; H(x)\cdot(x-\sigma-\hat{\gamma})\,(x-\sigma+\hat{\gamma}), \\ H(x) \;=\; x^{m-2} + h_{m-3}x^{m-3} + \cdots + h_1x + h_0, \\ 1/\eta \;=\; \max\{|h_1/h_0|,\ |h_2/h_0|^{1/2},\ \ldots,\ |1/h_0|^{1/(m-2)}\}. \end{cases} \tag{4.5}$$

Expressing $c_0, c_1, c_2, \ldots$ by $h_0, h_1, h_2, \ldots$, we obtain $c_0 = (\sigma^2-\hat{\gamma}^2)h_0,\ c_1 = (\sigma^2-\hat{\gamma}^2)h_1 - 2\sigma h_0,\ c_j = (\sigma^2-\hat{\gamma}^2)h_j - 2\sigma h_{j-1} + h_{j-2}\ \ (j \ge 2)$. By these, we obtain

$$\frac{c_1}{c_0} = -\frac{2\sigma}{\sigma^2-\hat{\gamma}^2} + \frac{h_1}{h_0}, \qquad \frac{c_2}{c_0} = \frac{1}{\sigma^2-\hat{\gamma}^2} - \frac{2\sigma}{\sigma^2-\hat{\gamma}^2}\frac{h_1}{h_0} + \frac{h_2}{h_0}.$$

Solving σ and $\hat{\gamma}$ from these equations, we obtain

$$2\sigma = \gamma_1 + \gamma_2 = -\frac{C_1}{C_2}, \qquad 2\hat{\gamma} = \gamma_1 - \gamma_2 = \frac{\sqrt{C_1^2 - 4C_2}}{C_2}, \tag{4.6}$$

where

$$C_1 = \frac{c_1}{c_0} - \frac{h_1}{h_0}, \qquad C_2 = \frac{c_2}{c_0} - \frac{c_1}{c_0}\frac{h_1}{h_0} + \frac{h_1^2}{h_0^2} - \frac{h_2}{h_0}. \tag{4.7}$$

The roots of $C_2x^2 + C_1x + 1$ are γ_1 and γ_2. In fact, equalities in (4.6) give us

$$C_2 = \frac{1}{\sigma^2 - \hat{\gamma}^2} = \frac{1}{\gamma_1\gamma_2}, \qquad C_1 = \frac{-2\sigma}{\sigma^2 - \hat{\gamma}^2} = \frac{-\gamma_1-\gamma_2}{\gamma_1\gamma_2}. \tag{4.8}$$

The above relation on c_j and h_j $(j \ge 2)$ also gives us

$$\frac{c_j}{c_2} = \frac{(h_{j-2}/h_0) - 2\sigma(h_{j-1}/h_0) + (\sigma^2-\hat{\gamma}^2)(h_j/h_0)}{1 - 2\sigma(h_1/h_0) + (\sigma^2-\hat{\gamma}^2)(h_2/h_0)} \qquad (j \ge 3). \tag{4.9}$$

Lemma 2. *The following inequalities hold for* $|C_1|$, $|C_2|$ *and* $|C_1^2-4C_2|/|C_2|^2$:

$$\left\{\begin{array}{rcl} \dfrac{|c_1|}{|c_0|}-\dfrac{1}{\eta} \ \le & |C_1| & \le \ \dfrac{|c_1|}{|c_0|}+\dfrac{1}{\eta}, \\ \dfrac{|c_2|}{|c_0|}-\dfrac{1}{\eta}\dfrac{|c_1|}{|c_0|}-\dfrac{2}{\eta^2} \ \le & |C_2| & \le \ \dfrac{|c_2|}{|c_0|}+\dfrac{1}{\eta}\dfrac{|c_1|}{|c_0|}+\dfrac{2}{\eta^2}, \\ \dfrac{|c_1^2-4c_2c_0|-2|c_1c_0|/\eta-7|c_0|^2/\eta^2}{(|c_2|+|c_1|/\eta+2|c_0|/\eta^2)^2} \ \le & |C_1^2-4C_2|/|C_2|^2. & \end{array}\right. \tag{4.10}$$

Proof. Definition of η gives $|h_j/h_0|\le 1/\eta^j$ for $j=1,2,\dots$. From these inequalities and the equality $C_1^2-4C_4=(c_1/c_0)^2-4(c_2/c_0)+2(c_1/c_0)(h_1/h_0)-3(h_1/h_0)^2+4(h_2/h_0)$, we obtain the above inequalities easily. □

Assuming that $|\sigma|$ and $|\hat\gamma|$ are small enough, let us bound η/d.

Lemma 3. *So long as* $|\sigma|/\eta\ll 1$ *and* $|\sigma^2-\hat\gamma^2|/\eta^2\ll 1$, *the following inequalities hold:*

$$\frac{1-2|\sigma|/\eta-|\sigma^2-\hat\gamma^2|/\eta^2}{1+2|\sigma|/\eta+|\sigma^2-\hat\gamma^2|/\eta^2} \le \frac{\eta}{d} \le \frac{1+2|\sigma|/\eta+|\sigma^2-\hat\gamma^2|/\eta^2}{1-2|\sigma|/\eta-|\sigma^2-\hat\gamma^2|/\eta^2}. \tag{4.11}$$

Proof. We can bound the r.h.s. expression in (4.9), let it be R_j, by inequalities $|h_{j'}/h_0|\le 1/\eta^{j'}$ ($j'=j, j–1, j–2$). Then, the above bound is obtained by bounding $|c_j/c_0|=|R_j|^{1/j}$ by inequality $[(1+r)/(1-r)]^{(1/j)}\le(1+r)/(1-r)$ which is valid for any integer $j>0$ and any real number r such that $0<r<1$.

We note that there exists a positive integer j'' such that $1/\eta=|h_{j''}/h_0|^{1/j''}$. For $j=j''+2$, we obtain $|(h_{j''}/h_0)-2\sigma(h_{j''+1}/h_0)+(\sigma^2-\hat\gamma^2)(h_{j''+2}/h_0)|\ge(1/\eta^{j''})\cdot(1-2|\sigma|/\eta-|\sigma^2-\hat\gamma^2|/\eta^2)$. Bounding $|c_j/c_2|$ in (4.9) by this inequality, and using $[(1-r)/(1+r)]^{(1/j)}\ge(1-r)/(1+r)$ which is valid for r such that $0<r<1$, we obtain the lower bound. □

We investigate the above inequalities by transforming $C(x)$ as

$$C(x)\mapsto C(\eta x)/h_0\eta^2 \overset{\text{def}}{=} C'(x).$$

Then, $H(x)$ is transformed as follows:

$$\left\{\begin{array}{c} H(x)\mapsto H(\eta x)/h_0 \overset{\text{def}}{=} H'(x), \\ H'(x)=h'_{m-2}x^{m-2}+\cdots+h'_1x+1, \\ \max\{|h'_{m-2}|,\ \dots,\ |h'_1|\}=1. \end{array}\right.$$

Applying formula (2.2) to $H'(x)$, we see that

$$|\gamma_j/\eta|\ge 1/2 \quad\text{or}\quad |\gamma_j|\ge\eta/2 \qquad (j\ge 3). \tag{4.12}$$

Therefore, η is a number showing how the roots $\gamma_3,\dots,\gamma_m$ are distributed. In fact, expressing the quantities in (4.10) by $c'_0=c_0/\eta^2$, $c'_1=c_1/\eta$, $c'_2=c_2$, $C'_1=C_1\eta$ and

$C_2' = C_2\eta^2$, we can remove η from inequalities in (4.10). Let the roots of $C'(x)$ be $\gamma_1', \dots, \gamma_m'$ ($\gamma_i' = \gamma_i/\eta$; $i = 1, \dots, m$). With γ_1', γ_2', we can rewrite (4.11) as follows:

$$\frac{1 - |\gamma_1'+\gamma_2'| - |\gamma_1'\gamma_2'|}{1 + |\gamma_1'+\gamma_2'| + |\gamma_1'\gamma_2'|} \leq \frac{\eta}{d} \leq \frac{1 + |\gamma_1'+\gamma_2'| + |\gamma_1'\gamma_2'|}{1 - |\gamma_1'+\gamma_2'| - |\gamma_1'\gamma_2'|}. \tag{4.13}$$

So long as γ_1' and γ_2' are small enough, the above inequalities are fairly accurate.

4.3. A Formula for the Minimum Root Separation

A fairly accurate upper bound of $\alpha_{\rm c}'$ is currently not obtained, although we have an order estimation by Proposition 3. Therefore, we set the following condition on c_2, c_1, c_0:

$$|c_1/c_2|^2 < |c_0/c_2| \iff |c_1|^2 < |c_2c_0|. \tag{4.14}$$

Since the origin has been moved to $\alpha_{\rm c}'$, this condition will be well satisfied. If the condition is not satisfied, we shift the origin slightly to satisfy the condition. With the above condition, we define e as follows:

$$e \overset{\rm def}{=} \max\{|c_1/c_2|, |c_0/c_2|^{1/2}\} = |c_0/c_2|^{1/2}. \tag{4.15}$$

Note that, if we transform $C(x)$ to a regularized form by $C(x) \mapsto \bar{C}(x) = C(dx)/c_2d^2 = \bar{c}_m x^m + \cdots + x^2 + \bar{c}_1 x + \bar{c}_0$ and define $\bar{e}$ as $\bar{e} = \min\{|\bar{c}_1|, |\bar{c}_0|^{1/2}\}$, then we have $\bar{e} = e/d$.

Theorem 6. *Let $\bar{R}_{\rm in}$ be the same as that in Theorem* 1*, with $\bar{e}$ replaced by e/d. Let $\bar{R}_{\rm in}$ be such that the polynomial $y^3 - (1-2\bar{R}_{\rm in})y^2 + \bar{R}_{\rm in}(2+\bar{R}_{\rm in})y + \bar{R}_{\rm in}^2$ has three real roots, and the largest root (a little smaller than 1) be $\eta_{\rm min}/d$ ($\eta_{\rm min}$ is a lower bound of η). If $e/d < 0.03$ as well as $|c_1|^2 < |c_2c_0|$ then the following inequality holds:*

$$|\gamma_1 - \gamma_2|^2 > \frac{|c_1^2 - 4c_2c_0| - 2|c_1c_0|/\eta_{\rm min} - 7|c_0|^2/\eta_{\rm min}^2}{(|c_2| + |c_1|/\eta_{\rm min} + 2|c_0|/\eta_{\rm min}^2)^2}. \tag{4.16}$$

Proof. We note that if $e/d < 0.03$ then $\bar{R}_{\rm in} < 0.0621\cdots$ and the above cubic polynomial has three real roots. The regularization of $C(x)$ in (4.4) tells us that $|\gamma_1|, |\gamma_2| < \bar{R}_{\rm in}d$, hence $|\sigma|/\eta < \bar{R}_{\rm in}(d/\eta)$ and $|\sigma^2 - \hat{\gamma}^2|/\eta^2 < \bar{R}_{\rm in}^2(d/\eta)^2$. Then, by putting $y = \eta/d$, (4.11) becomes

$$\frac{y^2 - 2\bar{R}_{\rm in}y - \bar{R}_{\rm in}^2}{y^2 + 2\bar{R}_{\rm in}y + \bar{R}_{\rm in}^2} < y < \frac{y^2 + 2\bar{R}_{\rm in}y + \bar{R}_{\rm in}^2}{y^2 - 2\bar{R}_{\rm in}y - \bar{R}_{\rm in}^2}.$$

The l.h.s. of this inequality gives us a lower bound of η, and we obtain $\eta_{\rm min}$ as in the theorem. Replacing η by $\eta_{\rm min}$ in the bottom inequality in (4.10), we obtain (4.16). □

Corollary 5. *If $e/d \geq 0.03$ then we have the following inequality:*

$$|\gamma_1 - \gamma_2| \geq 0.03d \cdot \frac{\sqrt{3 - 2\xi - 7\xi^2}}{1 + \xi + 2\xi^2}, \quad \text{where} \quad \xi = 0.0442\cdots. \tag{4.17}$$

Proof. As a critical case that the close-root cluster of γ_1 and γ_2 are separated from the other roots and inequality (4.11) holds narrowly, we consider the case of $\bar{e} = e/d = 0.03$ (this number is pretty small compared with 1/9). In this case, we have $|c_1/c_2| \le 0.03d$, $|c_0/c_2| = (0.03d)^2$, $\bar{R}_{\text{in}} = 0.0621\cdots$, and $\eta_{\min}/d = 0.678\cdots$. ($\xi = \bar{e}d/\eta_{\min} = 0.0442\cdots$). We bound $|c_1^2 - 4c_2c_0|$ as $|c_1^2 - 4c_2c_0| \ge |4c_2c_0| - |c_1|^2$, then the r.h.s. of (4.16) is monotone increasing for $\bar{e} \in [0, 1]$. Hence, substituting the actual values to $|c_1/c_2|$ etc. in the r.h.s. of (4.16), we obtain (4.17). □

5. Discussions

Our study in this paper is restricted in that the close roots are assumed to form well-separated clusters and that only polynomials over **C** are treated. Developing an algebraic algorithm for separating close roots distributed arbitrarily is a challenging theme. A more challenging theme is to find a reasonable lower bound for the minimum root separation for polynomials over **Z**.

The separation algorithm presented in Sect. 3 will be very useful practically. The underlying idea is so simple and effective that it will be applied to various problems. In fact, collaborating with Terui, one of the present authors (T.S.) developed recently a numerical algorithm for computing the close roots in a cluster simultaneously and efficiently.

The formula in the corollary of Theorem 6 is practically not bad as the lower bound for the minimum root separation. However, our formulas are not expressed by the coefficients of the original polynomial $A(x)$ but undergone a repetition of separation of close-root clusters, and they are not elaborated yet. Furthermore, our theory is not complete in that we set the condition (4.14). We should develop a theory without such a condition.

References

[CH74] G.E. Collins and E. Horowitz: The minimum root separation of a polynomial. Math. Comp. **28** (1974), 589–597.

[Col01] G.E. Collins: Polynomial minimum root separation. J. Symb. Comp. **32** (2001), 467–473.

[CWZ02] R.M. Corless, S.M. Watt and L. Zhi: QR factoring to compute the GCD of univariate approximate polynomial. Preprint of Univ. Western Ontario, Canada, 2002.

[HS97] V. Hribernig and H.J. Stetter: Detection and validation of clusters of zeros of polynomials. J. Symb. Comp. **24** (1997), 667–681.

[IS04] D. Inaba and T. Sasaki: Certification of analytic continuation of algebraic functions. Proc. CASC2004, (Eds. V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov), 249–260. July 2004.

[Mig92] M. Mignotte: *Mathematics for Computer Algebra.* Springer-Verlag, New York, 1992, Ch. 4.

[OST97] H. Ohsako, H. Sugiura and T. Torii: A stable extended algorithm for generating polynomial remainder sequence (in Japanese). Trans. of JSIAM (Japan Society for Indus. Appl. Math.) **7** (1997), 227–255.

[Pan95] V.Y. Pan: Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros. Proc. 27th Ann. ACM Symp. on Theory of Computing, 741–750. ACM Press, New York, 1995.

[Pan96] V.Y. Pan: Optimal and nearly optimal algorithms for approximating complex zeros. Computers & Math. (with Applications) **31** (1996), 97–138.

[Pan01] V.Y. Pan: Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. Proc. ISSAC 2001, (Ed. J.R. Sendra), 253–267. ACM Press, 2001.

[Sas03] T. Sasaki: The subresultant and clusters of close roots. Proc. ISSAC 2003, (Ed. B. Mourrain), 232–239. ACM Press, 2003.

[SN89] T. Sasaki and M-T. Noda: Approximate square-free decomposition and rootfinding of ill-conditioned algebraic equations. J. Inf. Process. **12** (1989), 159–168.

[SS89] T. Sasaki and M. Sasaki: Analysis of accuracy decreasing in polynomial remainder sequence with floating-point number coefficients. J. Inf. Process. **12** (1989), 159–168.

[SS97] T. Sasaki and M. Sasaki: Polynomial remainder sequence and approximate GCD. SIGSAM Bulletin **31** (1997), 4–10.

[SST92] T. Sakurai, H. Sugiura and T. Torii: Numerical factorization of polynomial by rational Hermite interpolation. Numerical Algorithms **3** (1992), 411–418.

[ST02] T. Sasaki and A. Terui: A formula for separating small roots of a polynomial. ACM SIGSAM Bulletin **36** (2002), 19–29.

[TS00] A. Terui and T. Sasaki: "Approximate zero-points" of real univariate polynomial with large error terms. IPSJ Journal (Information Processing Society of Japan) **41** (2000), 974–989.

[WH90] X.H. Wang and D.F. Han: On dominating sequence method in the point estimate and Smale theorem. Sci. China (Series A) **33** (1990), 135–144.

[Yak00] J.C. Yakoubsohn: Finding a cluster of zeros of univariate polynomials. J. Complexity **16** (2000), 603–636.

[Zen03] Z. Zeng: A method computing multiple roots of inexact polynomials. Proc. ISSAC 2003, (Ed. B. Mourrain), 166–272. ACM Press, 2003.

Tateaki Sasaki
Institute of Mathematics
University of Tsukuba
Tsukuba-shi, Ibaraki 305-8571, Japan
e-mail: `sasaki@math.tsukuba.ac.jp`

Fujio Kako
Department of Information and Computer Sciences
Nara Women's University
Nara-shi, Nara 630-8506, Japan
e-mail: `kako@ics.nara-wu.ac.jp`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 167–184
© 2007 Birkhäuser Verlag Basel/Switzerland

On the Location of Zeros of an Interval Polynomial

Hiroshi Sekigawa and Kiyoshi Shirayanagi

Abstract. For an interval polynomial F, we provide a rigorous method for deciding whether there exists a polynomial in F that has a zero in a prescribed domain D. When D is real, we show that it is sufficient to examine a finite number of polynomials. When D is complex, we assume that the boundary C of D is a simple closed curve of finite length and C is represented by a piecewise rational function. The decision method uses the representation of C and the property that a polynomial in F is of degree one with respect to each coefficient regarded as a variable. Using the method, we can completely determine the set of real numbers that are zeros of a polynomial in F. For complex zeros, we can obtain a set X that contains the set $Z(F)$, which consists of all the complex numbers that are zeros of a polynomial in F, and the difference between X and $Z(F)$ can be as small as possible.

Mathematics Subject Classification (2000). Primary 12D10, 26C10, 30C15.

Keywords. Interval polynomial, polynomial, zero, convex set, edge theorem.

1. Introduction

There are two premises for incorporating numeric or approximate computation in symbolic computation. One is that we know the exact values but use approximate computation for efficiency. An example is the theory of stabilizing algebraic algorithms [11, 12, 13]. The other is that inexact values are given.

In this article, we consider problems on the latter premise. That is, we treat the problems regarding zeros of real polynomials with perturbations. More precisely, let $[l_j, h_j] \subset \mathbb{R}$ be bounded closed intervals for $0 \le j \le d$. We consider the following types of problems.

- Does there exist a polynomial $f = a_d x^d + \cdots + a_0$ such that each a_j lies in the interval $[l_j, h_j]$ and f has a zero in the prescribed real (or complex) domain?
- What is the union of the sets of (real) zeros of polynomials $f = a_d x^d + \cdots + a_0$ such that each coefficient a_j lies in the interval $[l_j, h_j]$?

A similar but a slightly different problem on real zeros is treated in [5]. For a given real polynomial f that has no real zero, [5] provides an algorithm that finds the nearest real polynomial to f in the infinity norm among polynomials having a real zero. For complex domains, similar problems have been studied for complex polynomials (see, for example, [9, 4, 6, 2]). In such studies, complex perturbations are considered, and this is quite natural, since coefficients are complex. As described in [9, 7, 14], these studies can be viewed and understood in the common framework of fundamental observation from linear algebra.

Considering applications, we are very interested in the above types of problems since coefficients of polynomials may contain errors. For real polynomials, it is natural to consider only real perturbations, since in many practical examples real coefficients are obtained through measurements or observations and the errors are also real numbers. The methods in this article do not assume that perturbations are small except in several cases where small perturbations must be assumed so that the leading coefficient does not vanish.

It is also natural to consider only real zeros for many applications. Therefore, we treat real zeros in the first half and complex zeros in the second half. For real zeros, we provide a rigorous method for determining whether there exists a polynomial whose coefficients lie in the intervals $[l_j, h_j]$ and whose zero lies in the prescribed interval. We show that it is enough to examine only a finite number of polynomials. For complex zeros, we provide a rigorous method that first follows [9, 7, 14] but in the end differs from them because the perturbations are real. This type of research has already been carried out in control theory [3] and some results, such as Kharitonov's Theorem [8] and the Edge Theorem [1], have been obtained. The Edge Theorem is closely related to our main results on complex zeros; therefore, we will describe the relation between them.

This article is organized as follows. Section 2 introduces the notion of interval polynomials that can describe a set of polynomials with perturbations. Section 3 describes the decision method for real zeros. Section 4 describes the decision principle and computation methods for complex zeros and the relation with the Edge Theorem. Finally, Section 5 mentions future directions.

2. Definitions and Notations

In this section, we introduce interval polynomials to describe a set of polynomials with perturbations and pseudozeros to describe zeros of interval polynomials.

Definition 1 (Interval polynomials). For $1 \le j \le n$, let $e_j(x)$ be a monic polynomial in $\mathbb{R}[x]$ and $A_j = [l_j, h_j] \subset \mathbb{R}$ be a bounded closed interval. The set of polynomials

$$\left\{ \sum_{j=1}^{n} a_j e_j(x) \,\middle|\, a_j \in A_j \right\} \tag{$*$}$$

is said to be an interval polynomial. A_j is said to be an interval coefficient.

For simplicity, the set described by $(*)$ may be denoted as follows:

$$A_1e_1(x) + A_2e_2(x) + \cdots + A_ne_n(x).$$

Note that an interval polynomial F is a convex set from the definition.

Definition 2 (Pseudozeros). Let F be an interval polynomial. We define a point $c \in \mathbb{C}$ as a pseudozero of F if and only if there exists $f \in F$ such that $f(c) = 0$. We write all pseudozeros of F as $Z(F)$. A pseudozero c of F is said to be a real pseudozero if c is real. We write all real pseudozeros of F as $Z_{\mathbb{R}}(F)$.

When computing, we restrict the real and the imaginary parts of numbers to rational numbers or real algebraic numbers and use exact computation unless mentioned otherwise.

3. Deciding the Set of Real Pseudozeros

In this section, for an interval polynomial F, we provide a method for determining real pseudozeros of F. The fundamental tool is a method for determining whether there exists a polynomial $f \in F$ such that f has a zero in a given closed interval $D = [d_1, d_2]$ in $\mathbb{R}$. When $d_1 = d_2$, this can be determined by using interval arithmetic with exact computation for endpoints. When $d_1 < d_2$, the following lemma is the fundamental tool.

Lemma 1. *Let F be an interval polynomial as described by $(*)$. Suppose that every e_j has no zero in the interior of the interval D. Then, each e_j is always positive or negative in the interior of D. We denote by P the set of all indices j such that $e_j > 0$ and by N the set of all indices j such that $e_j < 0$. We put*

$$f_l(x) = \sum_{j\in P} l_je_j(x) + \sum_{j\in N} h_je_j(x), \qquad f_h(x) = \sum_{j\in P} h_je_j(x) + \sum_{j\in N} l_je_j(x).$$

Then, two polynomials f_l and f_h belong to F.

1. *If at least one of $f_l(d_1)$, $f_l(d_2)$, $f_h(d_1)$ and $f_h(d_2)$ is 0, or there exists a pair with opposite signs, then there exists $f \in F$ such that f has a zero in D.*
2. *When $f_l(d_1)$, $f_l(d_2)$, $f_h(d_1)$, $f_h(d_2) > 0$, there exists $f \in F$ such that f has a zero in D if and only if f_l has a zero in D.*
3. *When $f_l(d_1)$, $f_l(d_2)$, $f_h(d_1)$, $f_h(d_2) < 0$, there exists $f \in F$ such that f has a zero in D if and only if f_h has a zero in D.*

Proof. Note that $f_l(c) \le f(c) \le f_h(c)$ hold for any $f \in F$ and any $c \in D$.

First we prove Case 1. When one of $f_l(d_j)$ and $f_h(d_j)$ is 0, the statement is clear. If $f_l(d_1)f_l(d_2) < 0$ (resp. $f_h(d_1)f_h(d_2) < 0$), then the intermediate value theorem implies the statement. Suppose that $f_l(d_1)f_l(d_2) > 0$ and $f_h(d_1)f_h(d_2) > 0$ (see Fig. 1). Then, the sign of $f_l(d_1)$ and that of $f_h(d_1)$ should be opposite; otherwise all of the signs of $f_l(d_j)$ and $f_l(d_j)$ are the same. Put

$$t = \frac{f_h(d_1)}{f_h(d_1) - f_l(d_1)}.$$

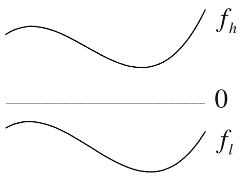


FIGURE 1. An example of the case $f_l(d_1)f_l(d_2),\ f_h(d_1)f_h(d_2) > 0$.

Then, the polynomial $g = (1-t)f_h + tf_l$ is in F and has a zero at d_1.

Since the proofs for Cases 2 and 3 are similar, we only show that of the former. Suppose that a polynomial $f \in F$ has a zero at c in the interval $[d_1, d_2]$. Then, $d_1 < c$ because $f_l(c) \le f(c) = 0$ and $f_l(d_1) > 0$. Therefore, the intermediate value theorem implies that f_l has a zero in the interval $[d_1, c]$. □

Remark 1. Arguments similar to the proof of Lemma 1 are valid for intervals $(-\infty, d_2]$, $[d_1, \infty)$, and $(-\infty, \infty)$ if no zero of e_j exists in the interior.

Remark 2. Under the same assumption of Lemma 1, every $f \in F$ has a zero in D if and only if both f_l and f_h have a zero in D.

Using Lemma 1, we can determine the real pseudozeros as follows.

Theorem 1. *Let F be an interval polynomial as described by* $(*)$. *Let all of the distinct real zeros of $\prod_{j=1}^{n} e_j$ be $\alpha_1 < \alpha_2 < \cdots < \alpha_m$. We make intervals $D_0 = (-\infty, \alpha_1]$, $D_k = [\alpha_k, \alpha_{k+1}]$ $(1 \le k \le m-1)$ and $D_m = [\alpha_m, \infty)$.*

For the interval D_k, we denote the polynomials corresponding to $f_l(x)$ and $f_h(x)$ described in Lemma 1 *by $f_{k,l}(x)$ and $f_{k,h}(x)$. Then, we have*

$$Z_{\mathbb{R}}(F) = \bigcup_{k=0}^{m} \left\{ c \in D_k \;\middle|\; f_{k,l}(c) \le 0 \le f_{k,h}(c) \right\}.$$

Proof. The inequalities $f_{k,l}(c) \le f(c) \le f_{k,h}(c)$ hold for any $c \in D_k$, $f \in F$ and $(1-t)f_{k,l} + tf_{k,h} \in F$ for any t $(0 \le t \le 1)$. These facts imply the statement. □

Corollary 1. *$Z_{\mathbb{R}}(F)$ is the union of a finite number* (*possibly zero*) *of closed intervals whose types are as follows:*

- *A closed interval $[\alpha, \beta]$, where α and β are zeros of $f_{k,l}$ or $f_{k,h}$.*
- *A closed interval $(-\infty, \alpha]$ or $[\alpha, \infty)$, where α is a zero of either $f_{k,l}$ or $f_{k,h}$.*
- *The whole real numbers $\mathbb{R}$.*

When the degrees of all polynomials in F are equal, only the first type appears.

Proof. Let m be the number of the distinct real zeros of $\prod_{j=1}^{n} e_j$. We take all of the distinct zeros of $f_{k,l}(x)$ and $f_{k,h}(x)$ in D_k described in Theorem 1 and denote them by $\beta_{k,1} < \beta_{k,2} < \cdots < \beta_{k,n(k)}$. Then, the signs of $f_{k,l}(x)$ and $f_{k,h}(x)$ do not change in the interval $(\beta_{k,p}, \beta_{k,p+1})$. The signs also do not change in the interval $(-\infty, \beta_{0,1})$ for $k = 0$ nor in the interval $(\beta_{m,n(m)}, \infty)$ for $k = m$.

Take every interval on which $f_{k,l}(x)$ is negative and $f_{k,h}(x)$ is positive and make it closed by adding the endpoints that are zeros of $f_{k,l}(x)$ or $f_{k,h}(x)$. $Z_{\mathbb{R}}(F)$ is the union of all such closed intervals. □

Example 1 (Lagrange interpolation). In the Lagrange interpolation for m points $a_1 < a_2 < \cdots < a_m$, each monic polynomial $e_j(x)$ is represented as follows:

$$e_j(x) = \prod_{k\neq j}(x - a_k).$$

We can easily compute real pseudozeros since $\{\, x \in \mathbb{R} \mid e_j(x) = 0 \,\}$=$\{\, a_k \mid k \neq j \,\}$.

Remark 3. The converse of the last part of Corollary 1 is not true. When the degrees are not constant, all types of intervals in Corollary 1 may appear. Consider the two monic polynomials $e_1(x)$ and $e_2(x)$:

$$e_1(x) = x^4 - 5, \qquad e_2(x) = x^4 - x^2.$$

We define three interval polynomials $F(x) \subset G(x) \subset H(x)$ as follows:

$$\begin{aligned} F(x) &= [1,\ 1.5]e_1(x) + [-1,\ -0.5]e_2(x), \\ G(x) &= [1,\ 1.5]e_1(x) + [-1.5,\ -0.5]e_2(x), \\ H(x) &= [0,\ 1.5]e_1(x) + [-1.5,\ 0]e_2(x). \end{aligned}$$

Note that both $e_1(x) - e_2(x)$ and $e_1(x) - e_2(x)/2$ belong to F $(\subset G \subset H)$, and $\deg(e_1(x) - e_2(x)) = 2$ and $\deg(e_1(x) - e_2(x)/2) = 4$. As described below, we have

$$\begin{aligned} Z_{\mathbb{R}}(F) &= \left[-\sqrt{5}, -\sqrt{5/2}\right] \cup \left[\sqrt{5/2}, \sqrt{5}\right], \\ Z_{\mathbb{R}}(G) &= \left(-\infty, -\sqrt{5/2}\right] \cup \left[\sqrt{5/2}, \infty\right), \\ Z_{\mathbb{R}}(H) &= \mathbb{R}. \end{aligned}$$

Real zeros of e_1 are $\pm\sqrt[4]{5}$ and real zeros of e_2 are 0 and ± 1. Since $f(-x) = f(x)$ for any $f \in H$, it is sufficient that we examine only in the interval $[0, \infty)$. We divide $[0, \infty)$ into three intervals: $[0, 1]$, $[1, \sqrt[4]{5}]$ and $[\sqrt[4]{5}, \infty)$. The polynomial e_1 is negative in $[0, \sqrt[4]{5})$ and positive in $(\sqrt[4]{5}, \infty)$. The polynomial e_2 is negative in $(0, 1)$ and positive in $(1, \infty)$.

First we examine G since it is clear that $Z_{\mathbb{R}}(H) = \mathbb{R}$. In the interval $[0, 1]$,

$$g_h(x) = e_1(x) - \frac{3}{2}e_2(x) = -\frac{1}{2}(x^4 - 3x^2 + 10) < 0.$$

In the interval $[1, \sqrt[4]{5}]$,

$$g_h(x) = e_1(x) - \frac{1}{2}e_2(x) = \frac{1}{2}(x^4 + x^2 - 10) < 0.$$

In the interval $[\sqrt[4]{5}, \infty)$,

$$\begin{aligned} g_l(x) &= e_1(x) - \frac{3}{2}e_2(x) = -\frac{1}{2}(x^4 - 3x^2 + 10) < 0, \\ g_h(x) &= \frac{3}{2}e_1(x) - \frac{1}{2}e_2(x) = x^4 + \frac{1}{2}x^2 - \frac{15}{2} = \frac{1}{2}(2x^2 - 5)(x^2 + 3). \end{aligned}$$

Therefore, the set of all real pseudozeros $Z_{\mathbb{R}}(G)$ is $(-\infty, -\sqrt{5/2}] \cup [\sqrt{5/2}, \infty)$, that is, the union of two unbounded closed intervals.

To determine $Z_{\mathbb{R}}(F)$, it is sufficient to examine only the interval $[\sqrt[4]{5}, \infty)$ since F is a subset of G. Now

$$\begin{aligned} f_l(x) &= e_1(x) - e_2(x) = x^2 - 5, \\ f_h(x) &= g_h(x) = \frac{1}{2}(2x^2 - 5)(x^2 + 3). \end{aligned}$$

Therefore, the set of all real pseudozeros $Z_{\mathbb{R}}(F)$ is $[-\sqrt{5}, -\sqrt{5/2}] \cup [\sqrt{5/2}, \sqrt{5}]$, that is, the union of two bounded closed intervals. Note that $\sqrt[4]{5} < \sqrt{5/2}$.

Next, we apply Theorem 1 to Wilkinson's famous example.

Example 2 (Wilkinson). Put $e_1(x) = \prod_{j=1}^{20}(x - j)$ and $e_2(x) = x^{19}$. We consider the following two interval polynomials $F(x) \subset G(x)$:

$$F(x) = e_1(x) + [-2^{-23}, 0]e_2(x), \qquad G(x) = e_1(x) + [-2^{-30}, 0]e_2(x).$$

The "endpoint" polynomial $e_1 - 2^{-23}e_2$ of F is Wilkinson's original example.

First, we consider $F(x)$. Since the signs of e_1 and e_2 do not change in the region $x < 0$, the polynomials f_l and f_h for $x \leq 0$ are as follows:

$$f_l(x) = e_1(x), \qquad f_h(x) = e_1(x) - 2^{-23}e_2(x).$$

Since $0 < f_l(x)$ for $x \leq 0$, there is no pseudozero in the region.

The interval coefficient of e_1 consists of one point and $0 \leq e_2(x)$ for $0 \leq x$. Therefore, for $0 \leq x$,

$$f_l(x) = e_1(x) - 2^{-23}e_2(x), \qquad f_h(x) = e_1(x).$$

The number of real zeros of f_l is 10 and there is no multiple root. We denote by $\alpha_1 < \alpha_2 < \cdots < \alpha_{10}$ the real zeros. They lie in the intervals as described below:

$$\begin{array}{ll} \alpha_1 \in (1 - 10^{-24},\ 1 - 10^{-25}), & \alpha_2 \in (2 + 10^{-18},\ 2 + 10^{-17}), \\ \alpha_3 \in (3 - 10^{-12},\ 3 - 10^{-13}), & \alpha_4 \in (4 + 10^{-10},\ 4 + 10^{-9}), \\ \alpha_5 \in (5 - 10^{-7},\ 5 - 10^{-8}), & \alpha_6 \in (6 + 10^{-6},\ 6 + 10^{-5}), \\ \alpha_7 \in (6.999,\ 6.9999), & \alpha_8 \in (8.001,\ 8.01), \\ \alpha_9 \in (8.9,\ 8.99), & \alpha_{10} \in (20.1,\ 21). \end{array}$$

From the above inequalities,

$$\begin{aligned} \{\, c \in \mathbb{R} \mid 0 \leq c,\ 0 \leq f_h(c) \,\} &= [0,1] \cup [2,3] \cup [4,5] \cup [6,7] \cup [8,9] \cup [10,11] \\ &\quad \cup [12,13] \cup [14,15] \cup [16,17] \cup [18,19] \cup [20,\infty), \\ \{\, c \in \mathbb{R} \mid 0 \leq c,\ f_l(c) \leq 0 \,\} &= [\alpha_1,\alpha_2] \cup [\alpha_3,\alpha_4] \cup [\alpha_5,\alpha_6] \cup [\alpha_7,\alpha_8] \cup [\alpha_9,\alpha_{10}]. \end{aligned}$$

Therefore, the set of all real pseudozeros for F is as follows:

$$\begin{aligned} Z_{\mathbb{R}}(F) &= [\alpha_1, 1] \cup [2, \alpha_2] \cup [\alpha_3, 3] \cup [4, \alpha_4] \cup [\alpha_5, 5] \cup [6, \alpha_6] \cup [\alpha_7, 7] \cup [8, \alpha_8] \\ &\quad \cup [\alpha_9, 9] \cup [10, 11] \cup [12, 13] \cup [14, 15] \cup [16, 17] \cup [18, 19] \cup [20, \alpha_{10}]. \end{aligned}$$

Similar arguments hold for G. It is clear that $Z_{\mathbb{R}}(G) \cap (-\infty, 0]$ is empty. For $0 \le x$,

$$g_l(x) = e_1(x) - 2^{-30} e_2(x), \qquad g_h(x) = e_1(x).$$

The number of real zeros of $g_l(x)$ is 14 and there is no multiple root. We denote by $\beta_1 < \beta_2 < \cdots < \beta_{14}$ the real zeros. They lie in the intervals as described below:

$$\begin{array}{ll}
\beta_1 \in (1 - 10^{-26},\ 1 - 10^{-27}), & \beta_2 \in (2 + 10^{-20},\ 2 + 10^{-19}), \\
\beta_3 \in (3 - 10^{-14},\ 3 - 10^{-15}), & \beta_4 \in (4 + 10^{-12},\ 4 + 10^{-11}), \\
\beta_5 \in (5 - 10^{-9},\ 5 - 10^{-10}), & \beta_6 \in (6 + 10^{-8},\ 6 + 10^{-7}), \\
\beta_7 \in (7 - 10^{-5},\ 7 - 10^{-6}), & \beta_8 \in (8 + 10^{-5},\ 8 + 10^{-4}), \\
\beta_9 \in (8.999,\ 8.9999), & \beta_{10} \in (10.001,\ 10.01), \\
\beta_{11} \in (10.9,\ 10.99), & \beta_{12} \in (12.3,\ 12.4), \\
\beta_{13} \in (12.4,\ 12.5), & \beta_{14} \in (20.01,\ 20.1).
\end{array}$$

From the above inequalities,

$$\begin{aligned}
\{\, c \in \mathbb{R} \mid 0 \le c,\ 0 \le g_h(c) \,\} &= [0,1] \cup [2,3] \cup [4,5] \cup [6,7] \cup [8,9] \cup [10,11] \\
&\quad \cup [12,13] \cup [14,15] \cup [16,17] \cup [18,19] \cup [20,\infty), \\
\{\, c \in \mathbb{R} \mid 0 \le c,\ g_l(c) \le 0 \,\} &= [\beta_1,\beta_2] \cup [\beta_3,\beta_4] \cup [\beta_5,\beta_6] \cup [\beta_7,\beta_8] \cup [\beta_9,\beta_{10}] \\
&\quad \cup [\beta_{11},\beta_{12}] \cup [\beta_{13},\beta_{14}].
\end{aligned}$$

Therefore, the set of all real pseudozeros for G is as follows:

$$\begin{aligned}
Z_{\mathbb{R}}(G) &= [\beta_1,1] \cup [2,\beta_2] \cup [\beta_3,3] \cup [4,\beta_4] \cup [\beta_5,5] \cup [6,\beta_6] \cup [\beta_7,7] \\
&\quad \cup [8,\beta_8] \cup [\beta_9,9] \cup [10,\beta_{10}] \cup [\beta_{11},11] \cup [12,\beta_{12}] \cup [\beta_{13},13] \\
&\quad \cup [14,15] \cup [16,17] \cup [18,19] \cup [20,\beta_{14}].
\end{aligned}$$

4. Deciding the Location of Pseudozeros

In this section, first we describe a principle for deciding the location of pseudozeros. Let F be an interval polynomial as described by $(*)$ and D be a domain in $\mathbb{C}$. We consider the following problem.

Problem 1. Does there exist a pseudozero of F in D?

Below, we assume that D is a closed domain in $\mathbb{C}$ whose boundary C is a simple curve. When D is not bounded, we further assume that the degree of $f \in F$ is constant. Since the domain D is not bounded when C is not a closed curve, from the above assumption on the degree we can construct a new closed domain $D' \subset D$ such that the following conditions are satisfied.

- The boundary of D' is a simple and closed curve.
- $Z(F) \cap D = Z(F) \cap D'$.

Therefore, we can assume that C is a simple and closed curve. Furthermore, we assume the following conditions.

Condition 1. C is of finite length and $C = \cup_{k=1}^{K} C_k$ ($K < \infty$), where each C_k is expressed by an injective function as

$$\varphi_k(s) + i\psi_k(s), \qquad s \in S_k.$$

Here $\varphi_k(s)$, $\psi_k(s) \in \mathbb{Q}(s)$ and S_k is either of type $[a, b]$, $[a, \infty)$, $(-\infty, b]$ or $\mathbb{R}$.

First, we reduce the problem to examine zeros on C. Second, we reduce it to examine polynomials whose coefficients are the endpoints of the interval coefficients with at most one exception.

4.1. Preliminaries

Take a polynomial $f_0 \in F$. We can determine whether f_0 has a zero on C using Sturm's algorithm, the sign variation method, or some other improved algorithm, and whether it has a zero in the interior of D using the argument principle when it has no zero on C. If f_0 has no zero in D, then Problem 1 is equivalent to asking whether there exists a pseudozero of F on C.

Proposition 1. *Suppose that a polynomial $f_0 \in F$ has no zero in D. When D is unbounded we assume that degrees of all polynomials in F are equal. Then, the following two conditions are equivalent.*

1. *There exists a polynomial $f \in F$ that has a zero in D.*
2. *There exists a polynomial $f \in F$ that has a zero on C.*

Proof. It is sufficient to prove that the first condition implies the second condition. Assume that f has a zero in D but no zero on C. Let g_t be $(1-t)f_0 + tf$. Then, $g_0 = f_0$, $g_1 = f$ and $g_t \in F$ for any t ($0 \le t \le 1$). We prove the statement by contradiction. Suppose that every g_t ($0 \le t \le 1$) has no zero on C.

When D is bounded, Rouché's theorem (see below) implies that the number of zeros of g_0 in D is equal to that of g_1. This contradicts the assumption.

When D is unbounded, the assumption that C is of finite length implies that the compliment D^c of D is bounded. Therefore, the number of zeros of g_0 in D^c is equal to that of g_1. Since $\deg g_0 = \deg g_1$, the number of zeros of g_0 in D is equal to that of g_1. This contradicts the assumption. □

The following is a version of Rouché's theorem.

Theorem 2 (Rouché's Theorem). *Let C be a simple closed curve of finite length in a domain $\Omega \subset \mathbb{C}$ and let the inside of C be in Ω. Suppose that $f(z)$ and $\varphi(z)$ are holomorphic on Ω and that $f(z) + t\varphi(z)$ has no zero on C for any t ($0 \le t \le 1$). Then, the number of zeros of $f(z)$ inside C is equal to that of $f(z) + \varphi(z)$.*

We provide a proof since this version is not described in standard textbooks.

Proof. The following inequality holds for $0 \le t_1 \le t_2 \le 1$:

$$|f(z) + t_2\varphi(z)| \le |f(z) + t_1\varphi(z)| + |t_2 - t_1||\varphi(z)|.$$

Let $m(t)$ be $\min_{z \in C}\{|f(z) + t\varphi(z)|\}$ and M be $\max_{z \in C}\{|\varphi(z)|\}$. Then,

$$m(t_2) \le |f(z) + t_1\varphi(z)| + |t_2 - t_1|M.$$

Therefore, the following inequality holds:

$$m(t_2) \le m(t_1) + |t_2 - t_1|M.$$

When we interchange t_1 and t_2, the resulting inequality also holds. Therefore,

$$|m(t_2) - m(t_1)| \le |t_2 - t_1|M.$$

This inequality implies that $m(t)$ is continuous in the interval $0 \le t \le 1$. From the hypothesis, $m(t) > 0$ holds for any t in $[0,1]$. Therefore, $m = \min_{0\le t\le 1}\{|m(t)|\}$ should be positive. Now, we denote the length of C by L and the maximum of $|f(z)\varphi'(z) - f'(z)\varphi(z))|$ on C by G. Let $N(t)$ be

$$\frac{1}{2\pi i}\int_C \frac{f'(z) + t\varphi'(z)}{f(z) + t\varphi(z)}dz.$$

Then, we have

$$|N(t_2) - N(t_1)| = \frac{1}{2\pi}\left|\int_C \frac{(t_1 - t_2)(f(z)\varphi'(z) - f'(z)\varphi(z))}{(f(z) + t_1\varphi(z))(f(z) + t_2\varphi(z))}dz\right| \le \frac{|t_1 - t_2|GL}{2\pi m^2},$$

which implies that $N(t)$ is continuous on the interval $0 \le t \le 1$. Therefore, the equality $N(0) = N(1)$ holds since $N(t)$ is a nonnegative integer for any t. □

4.2. Main Theorem

In this subsection, we prove the main theorem.

Theorem 3. *Let F be an interval polynomial as described by $(*)$ and D be a closed domain whose boundary C satisfies Condition 1. When D is unbounded we assume that degrees of all polynomials in F are equal. Suppose the following conditions.*

- *There exists a polynomial $f_0 \in F$ that does not have a zero in D.*
- *There exists a point $\alpha_0 \in C$ that is not a pseudozero of F.*

Then, the following two conditions are equivalent.

1. *There exists a polynomial $f \in F$ having a zero in D.*
2. *There exists a polynomial $f \in F$ such that f has a zero on C and the number of coefficients a_j of f that are not l_j or h_j is at most one.*

We prove the following lemma for the proof of Theorem 3.

Lemma 2. *Consider the following simultaneous equations with a parameter $z \in \mathbb{C}$,*

$$\begin{cases} a_1(z)x + b_1(z)y &= c_1(z), \\ a_2(z)x + b_2(z)y &= c_2(z), \end{cases} \tag{1}$$

where $a_j(z)$, $b_j(z)$, $c_j(z)$ are continuous with respect to z. Let Γ be a simple curve of finite length whose two endpoints are z_0 and z_1 $(z_0 \ne z_1)$. Let $d(z)$ be the determinant

$$\begin{vmatrix} a_1(z) & b_1(z) \\ a_2(z) & b_2(z) \end{vmatrix}.$$

Suppose the following conditions:

- $d(z_0) = 0$ *and* $d(z) \neq 0$ *for* $z \in \Gamma \setminus \{z_0\}$.
- *The solutions of* (1) *are bounded for* $z \in \Gamma \setminus \{z_0\}$.

Then, the simultaneous equations (1) *are indeterminate at* $z = z_0$.

Proof. For $z \in \Gamma \setminus \{z_0\}$ we put

$$n_x(z) = \begin{vmatrix} c_1(z) & b_1(z) \\ c_2(z) & b_2(z) \end{vmatrix}, \qquad n_y(z) = \begin{vmatrix} a_1(z) & c_1(z) \\ a_2(z) & c_2(z) \end{vmatrix}.$$

Then, the solution x and y of (1) can be represented as functions of z as follows:

$$x(z) = \frac{n_x(z)}{d(z)}, \qquad y(z) = \frac{n_y(z)}{d(z)}.$$

The functions $n_x(z)$ and $n_y(z)$ converge to 0 since $d(z)$ converges to 0 as z tends to z_0 on Γ and the solution of (1) is bounded in $\Gamma \setminus \{z_0\}$. Therefore,

$$n_x(z_0) = n_y(z_0) = 0, \tag{2}$$

since $n_x(z)$ and $n_y(z)$ are continuous. Furthermore, if $a_j(z_0) = b_j(z_0) = 0$ hold, then $c_j(z_0) = 0$. The reason is as follows. Since there exists a positive number M such that $|x(z)| \leq M$ and $|y(z)| \leq M$ hold for $z \in \Gamma \setminus \{z_0\}$, the inequality

$$|c_j(z)| = |a_j(z)x(z) + b_j(z)y(z)| \leq (|a_j(z)| + |b_j(z)|)M$$

holds for $z \in \Gamma \setminus \{z_0\}$. Since $(|a_j(z)| + |b_j(z)|)M$ converges to 0 when z tends to z_0, $c_j(z)$ converges to 0 and $c_j(z_0) = 0$ follows from the fact that $c_j(z)$ is continuous.

We prove the lemma by dividing it into three cases.

First, we prove the case $a_1(z_0) = a_2(z_0) = 0$. If $b_1(z_0) \neq 0$, then the second equation of (1) is equal to the first equation multiplied by $b_2(z_0)/b_1(z_0)$. Again, $b_1(z_0) \neq 0$ implies the conclusion. If $b_1(z_0) = 0$, then $c_1(z_0) = 0$ and the equations of (1) become the second equation only. Furthermore, if $b_2(z_0) = 0$, the second equation also vanishes.

Second, we prove the case $a_1(z_0) = 0$ and $a_2(z_0) \neq 0$. (The case $a_1(z_0) \neq 0$ and $a_2(z_0) = 0$ is similar.) The assumption $d(z_0) = 0$ implies that $b_1(z_0) = 0$. Therefore, $c_1(z_0) = 0$ and the second equation of (1) vanishes. Then, the assumption $a_2(z_0) \neq 0$ implies the conclusion.

The last case is that when both $a_1(z_0)$ and $a_2(z_0)$ are not 0. The assumption $d(z_0) = 0$ and (2) imply that the second equation of (1) is equal to the first equation multiplied by $a_2(z_0)/a_1(z_0)$ and $a_2(z_0) \neq 0$ implies the conclusion. □

The proof of the main theorem is as follows.

Proof. It is sufficient to prove that condition (1) implies condition (2) under the assumptions of the theorem.

From Proposition 1, there exists a polynomial $g \in F$ having a zero α on C.

If the number of coefficients of g that are not the endpoints of the interval coefficients is less than two, the proof is done.

Otherwise, we take two of them and write them as t_1 and t_2. Then, we can write the real part of $g(z)$ and the imaginary part of $g(z)$ as follows:

$$\operatorname{Re} g(z) = a_1(z)t_1 + b_1(z)t_2 + c_1(z), \qquad \operatorname{Im} g(z) = a_2(z)t_1 + b_2(z)t_2 + c_2(z),$$

where $a_j(z)$, $b_j(z)$, $c_j(z)$ are continuous functions with respect to z. The equation $g(\alpha) = 0$ is equivalent to the simultaneous equations

$$\begin{cases} a_1(z)t_1 + b_1(z)t_2 + c_1(z) = 0, \\ a_2(z)t_1 + b_2(z)t_2 + c_2(z) = 0. \end{cases} \tag{3}$$

We consider these to be the simultaneous equations of t_1 and t_2 with parameter z. If the determinant

$$\begin{vmatrix} a_1(z) & b_1(z) \\ a_2(z) & b_2(z) \end{vmatrix} \tag{4}$$

is 0 at $z = \alpha$, then we can move t_1 and t_2 as α is a zero, until either t_1 or t_2 reaches one of the end points of the interval coefficients.

If the determinant is not 0 at $z = \alpha$, the solutions t_1 and t_2 are continuous with respect to z whenever the determinant is not 0. Therefore, when we move z from α to α_0 on C, one of the following occurs.

(a) The determinant (4) is not 0, and either t_1 or t_2 reaches one of the endpoints of the interval coefficients.

(b) The determinant (4) is 0 at a point β.

If the determinant (4) is not 0, and t_1 and t_2 are in the interval coefficients as z tends to α_0, then from Lemma 2, the simultaneous equations (3) are indeterminate at $z = \alpha_0$. This contradicts the assumption that α_0 is not a pseudozero of F.

When case (a) occurs, we find that there exists $h \in F$ such that h has a zero on C and the number of coefficients of h that are not equal to the endpoints of the interval coefficients is less than that of g. If case (b) occurs, we can move t_1 and t_2 as (3) holds at $z = \beta$, until either t_1 or t_2 reaches the endpoints of the interval coefficients. That is, also in this case, we can find a polynomial $h \in F$ such that h has a zero on C and the number of coefficients of h that are not equal to the endpoints of the interval coefficients is less than that of g.

We apply this procedure repeatedly until condition (2) is satisfied. $\square$

4.3. Edge Theorem

Here, we describe the relation between Theorem 3 and the Edge Theorem [1].

In the Edge Theorem, the notion of a polynomial polytope appears, which is an extension of an interval polynomial. The set of polynomials represented as a convex combination of a given finite number of polynomials is said to be a polynomial polytope. For more details see textbooks on control theory.

Theorem 4 (Edge Theorem). *Suppose that a domain $D \subset \mathbb{C}$ satisfies the condition "any point in the compliment of D is on a path to infinity." Let F be a polynomial polytope. All zeros of any polynomial in F are contained in D if and only if all zeros of any exposed edge of F are contained in D.*

Remark 4. For an interval polynomial, an exposed edge is a subset of the set of polynomials whose coefficients are the endpoints of the interval coefficients except at most one coefficient. However, we should examine all such polynomials since there is no efficient way to find all exposed edges.

Suppose that Theorem 3 can be applied to a domain D and the Edge Theorem can be applied to the compliment D^c of D. Then, as described above, we can solve Problem 1 using Theorem 3. We can also solve Problem 1 by applying the Edge Theorem to D^c because the negation of the statement "there exists a polynomial $f \in F$ such that at least one zero of f belongs to D" is "all zeros of any polynomial in F belong to D^c." The computational cost when we use Theorem 3 is slightly high; determining whether there exists a zero in D for a given polynomial in F and whether there exists a polynomial f in F such that f has a zero at a given point on C are added.

However, the strong point of Theorem 3 over the Edge Theorem is that there exists a domain D such that Theorem 3 can be applied to D but the Edge Theorem cannot be applied to D^c. Closed disks and closed rectangles are such examples.

4.4. Computation Method

In this section, we describe the computation method using Theorem 3.

First, we show the method for deciding whether a given point on C is a pseudozero of an interval polynomial.

4.4.1. Polynomials Having a Zero at a Given Point of the Boundary. Let F be an interval polynomial as described by $(*)$. Then, we can write

$$F(x) = \left\{ \sum_{j=1}^{n} \{(h_j - l_j)t_j + l_j\}\, e_j(x) \;\middle|\; 0 \le t_j \le 1 \right\}.$$

Therefore, there exists a polynomial $f \in F$ such that f has a zero at a complex number α if and only if the equation

$$\sum_{j=1}^{n} \{(h_j - l_j)t_j + l_j\}\, e_j(\alpha) = 0$$

has a solution $0 \le t_j \le 1$ for all j. If (1) $\mathrm{Re}(h_j - l_j)e_j(\alpha) < 0$ or (2) $\mathrm{Re}(h_j - l_j)e_j(\alpha) = 0$ and $\mathrm{Im}(h_j - l_j)e_j(\alpha) < 0$, we replace t_j by $1 - t_j$ and we write the resulting equation as follows:

$$\sum_{j=1}^{n} a_j t_j = b. \tag{5}$$

Here, we take $\arg z$ for $z \in \mathbb{C}$ in the range $-\pi < \arg z \le \pi$. Therefore, the above substitution implies the inequalities $-\pi/2 < \arg a_j \le \pi/2$ for $a_j \ne 0$. We consider the problem that (5) has a solution $0 \le t_j \le 1$ for all j.

Lemma 3. *The set* $\{\sum_{j=1}^{n} a_j t_j \mid 0 \le t_j \le 1\}$ *is equal to the convex hull of the set* $\{\sum_{j=1}^{n} \varepsilon_j a_j \mid \varepsilon_j = 0,\, 1\}$.

Proof. It is clear that the set $\{\sum_{j=1}^{n} a_j t_j \mid 0 \le t_j \le 1\}$ contains the convex hull of the set $\{\sum_{j=1}^{n} \varepsilon_j a_j \mid \varepsilon_j = 0, 1\}$.

To show the latter contains the former, take any element $\beta = \sum_{j=1}^{n} a_j t_j$ in $\{\sum_{j=1}^{n} a_j t_j \mid 0 \le t_j \le 1\}$ and sort t_j in increasing order $0 \le t_{J(1)} \le t_{J(2)} \le \cdots \le t_{J(n)} \le 1$. Then,

$$\beta = \sum_{j=1}^{n} a_j t_j = \left(1 - t_{J(n)}\right) \cdot 0 + t_{J(1)} \sum_{j=1}^{n} a_{J(j)} + \sum_{k=2}^{n} \left\{ \left(t_{J(k)} - t_{J(k-1)}\right) \sum_{j=k}^{n} a_{J(j)} \right\}$$

and the equalities show that β is represented as a convex combinations of 0 and $\sum_{j=k}^{n} a_{J(j)}$ $(1 \le k \le n)$. □

Hence, we construct the convex hull of the set $V = \{\sum_{j=1}^{n} \varepsilon_j a_j \mid \varepsilon_j = 0 \text{ or } 1\}$ and determine whether b is in the convex hull. There are 2^n points in V in general, but the convex hull can be constructed efficiently: We can construct it by examining at most n points $a_1, a_2, \ldots, a_n$.

Theorem 5. *Let the set V be as above. First, sort $a_j \ne 0$ as the arguments in increasing order. Note that $-\pi/2 < \arg a_j \le \pi/2$ hold. If two or more points, say a_j, a_k, a_l, have the same argument, then we add them up together and replace a_j, a_k, a_l with the sum, and write the results as $p_1, p_2, \ldots, p_m$. Then, the vertices of the convex hull are, in counterclockwise order, $0, v_1, \ldots, v_{2m-1}$, where*

$$v_j = \begin{cases} \displaystyle\sum_{k=1}^{j} p_k & (1 \le j \le m), \\ \displaystyle\sum_{k=j-m+1}^{m} p_k & (m+1 \le j \le 2m-1). \end{cases}$$

We need a lemma for the proof. For $z_1, z_2 \in \mathbb{C}$, we define

$$d(z_1, z_2) = \begin{vmatrix} \operatorname{Re} z_1 & \operatorname{Re} z_2 \\ \operatorname{Im} z_1 & \operatorname{Im} z_2 \end{vmatrix}.$$

Then, the following lemma is clear.

Lemma 4. 1. *For any z, z_1 and $z_2 \in \mathbb{C}$ and $a \in \mathbb{R}$,*

$$d(z,z) = 0, \qquad d(z_1,z_2) = -d(z_2,z_1), \qquad d(az_1,z_2) = d(z_1,az_2) = a \cdot d(z_1,z_2),$$
$$d(z_1+z_2,z) = d(z_1,z) + d(z_2,z), \qquad d(z,z_1+z_2) = d(z,z_1) + d(z,z_2).$$

2. *When z_1 and z_2 are not 0 and $-\pi/2 < \arg z_1, \arg z_2 \le \pi/2$ hold, $d(z_1,z_2) > 0$ holds if and only if $\arg z_1 < \arg z_2$ holds, and $d(z_1,z_2) = 0$ holds if and only if $\arg z_1 = \arg z_2$ holds.*

Now, we prove Theorem 5.

Proof. For any j $(1 \le j \le 2m)$, it is sufficient to prove that an arbitrary point $\sum_{k=1}^{n} \varepsilon_k a_k$ is sitting at the left of or on the straight line from v_{j-1} to v_j (we put $v_0 = v_{2m} = 0$). To prove this, we introduce the following two statements.

- For any j and $a = \sum_{k=1}^{n} \varepsilon_k a_k$, the inequality $d(p_j, a - v_{j-1}) \geq 0$ holds.
- Any $a = \sum_{k=1}^{n} \varepsilon_k a_k$ satisfying $d(p_j, a - v_{j-1}) = 0$ lies between v_{j-1} and v_j.

First, we prove the first statement when $1 \leq j \leq m$. We divide $\sum_{k=1}^{n} \varepsilon_k a_k$ into three parts: S_1 consisting of a_k's whose arguments are less than $\arg p_j$, S_2 consisting of a_k's whose arguments are equal to $\arg p_j$, and S_3 consisting of a_k's whose arguments are greater than $\arg p_j$. Then, we have

$$d(p_j, a - v_{j-1}) = d(p_j, a - \sum_{k=1}^{j-1} p_k) = d(p_j, S_1 - \sum_{k=1}^{j-1} p_k) + d(p_j, S_2) + d(p_j, S_3).$$

From the definitions of S_2 and S_3, we have $d(p_j, S_2) = 0$ and $d(p_j, S_3) \geq 0$. Furthermore, the definition of S_1 implies

$$d(p_j, S_1) = d(p_j, \sum_k \varepsilon_k a_k) \geq d(p_j, \sum_{\arg a_k < \arg p_j} a_k) = d(p_j, \sum_{k=1}^{j-1} p_k).$$

Hence, the first statement is proved.

Next, we prove the second statement. Lemma 4 implies that the equality $d(p_j, a_k) = 0$ holds if and only if $a_k = 0$ or $\arg a_k = \arg p_j$ (The construction of p_j implies that $p_j \neq 0$). For the sum $a = \sum_{k=1}^{n} \varepsilon_k a_k$, we only add $a_k \neq 0$. The proof of the first statement implies that the equality $d(v_j - v_{j-1}, a) = 0$ holds if and only if the equality $\varepsilon_k = 1$ holds for k such that $\arg a_k < \arg p_j$ and the equality $\varepsilon_k = 0$ holds for k such that $\arg a_k > \arg p_j$. Therefore, the equality

$$a = v_{j-1} + \sum_{\arg a_k = \arg p_j} \varepsilon_k a_k$$

holds, and the equalities $v_j = v_{j-1} + p_j$ and

$$p_j = \sum_{\arg a_k = \arg p_j} a_k$$

imply the statement.

Similar arguments hold for $m + 1 \leq j \leq 2m$, considering that $v_j - v_{j-1} = -p_{j-m}$, by dividing $\sum_{k=1}^{n} \varepsilon_k a_k$ into S_1 consisting of a_k's whose arguments are less than $\arg p_{j-m}$, into S_2 consisting of a_k's whose arguments are equal to $\arg p_{j-m}$, and into S_3 consisting of a_k's whose arguments are greater than $\arg p_{j-m}$. □

4.4.2. Polynomials Having Zeros on the Boundary. Let F be an interval polynomial as described by $(*)$. For a polynomial $f = \sum_{j=1}^{n} a_j e_j \in F$, suppose that a_j is either l_j or h_j for $j \neq \lambda$. Here, we describe a method for determining whether we can make f have a zero on the segment $C_k \subset C$ by moving a_λ.

We write the representation of C_k as $\varphi_k(s) + i\psi_k(s)$, $s \in S_k$. For simplicity, writing a_λ as t and $[l_\lambda, h_\lambda]$ as $[l, h]$, we have

$$f(x) = t e_\lambda(x) + \sum_{j \neq \lambda} a_j e_j(x), \qquad t \in [l, h].$$

Substituting $\varphi_k(s) + i\psi_k(s)$ for x, we have

$$f(\varphi_k(s) + i\psi_k(s)) = u(s,t) + iv(s,t),$$

where u, $v \in \mathbb{Q}(s,t)$. Therefore, f has a zero on C_k for some $a_\lambda \in [l_\lambda, h_\lambda]$ if and only if the following simultaneous equations have a solution $s \in S_k$ and $l \leq t \leq h$.

$$\begin{cases} u(s,t) &= 0, \\ v(s,t) &= 0. \end{cases} \tag{6}$$

These equations are of degree one with respect to t. Therefore, solving $u = 0$ and $v = 0$, we write $t = T_1(s)$ and $t = T_2(s)$, where T_1, $T_2 \in \mathbb{Q}(s)$ (see Remark 5 below). Moreover, we put

$$T_1(s) - T_2(s) = \frac{P(s)}{Q(s)}, \tag{7}$$

where P, $Q \in \mathbb{Q}[s]$ and $\gcd(P,Q) = 1$.

Therefore, the problem is whether there exists a zero $\alpha \in S_k$ of P that satisfies $l \leq T(\alpha) \leq h$ by putting $T = T_1$ or T_2 (we can take either).

When $l = h$, we set

$$T(s) - l = \frac{P_l(s)}{Q_l(s)}, \tag{8}$$

where P_l, $Q_l \in \mathbb{Q}[s]$. Put $G_l = \gcd(P, P_l)$, we can solve Problem 1 by examining whether there exists a zero $\alpha \in S_k$ of G_l since the equations $T(\alpha) = l$ and $P(\alpha) = 0$ hold for any zero α of G_l, and $T(\beta) \neq l$ for any zero β of P/G_l.

When $l < h$, we compute (8) and

$$T(s) - h = \frac{P_h(s)}{Q_h(s)}, \tag{9}$$

where P_h, $Q_h \in \mathbb{Q}[s]$, $\gcd(P_h, Q_h) = 1$. Put $G_l = \gcd(P, P_l)$ and $G_h = \gcd(P, P_h)$. Then, G_l and G_h are relatively prime because $T(\alpha) = l$ for any zero α of G_l and $T(\beta) = h$ for any zero β of G_h. Therefore, we can divide zeros of P into three groups: the zeros of G_l, the zeros of G_h, and the zeros of $P/(G_l G_h)$.

We only need to examine real zeros since $S_k \subset \mathbb{R}$. For a real zero α of G_l, we examine whether $\alpha \in S_k$, and we can solve this using, for example, Sturm's algorithm (or some other efficient algorithm). We carry out a similar procedure for a real zero of G_h.

For a real zero α of $P/(G_l G_h)$, we examine whether $\alpha \in S_k$ and $l \leq T(\alpha) \leq h$. The former can be carried out using, for example, Sturm's algorithm. The latter can be carried out using approximate computation with error analysis, for example, by interval computation, under the assumption that we can raise the precision as high as desired since $T(\alpha)$ is not equal to l or h.

To summarize, for the rational function $\varphi_k(s) + i\psi_k(s)$ $(s \in S_k)$ that represents the segment C_k of C, the simultaneous equations in (6) determined by f and the range $[l, h]$ of t, we carry out the following computations.

- Noting that the equations in (6) are of degree one with respect to t, solving $u = 0$ and $v = 0$, we write $t = T_1(s)$ and $t = T_2(s)$, where T_1, $T_2 \in \mathbb{Q}(s)$ (see Remark 5 below).
- Compute (7).
- Put $T(s)$ as either $T_1(s)$ or $T_2(s)$ (we can use either).
- When $l = h$, compute (8). If $\gcd(P, P_l)$ has a zero in S_k, the answer to the question posed in Problem 1 is "Yes."
- When $l < h$, compute (8) and (9), and put $G_l = \gcd(P, P_l)$, $G_h = \gcd(P, P_h)$.
  If G_l or G_h has a zero in S_k, the answer is "Yes."
  If both G_l and G_h have no zero in S_k, when $P/(G_l G_h)$ has a zero α that is in S_k and $l < T(\alpha) < h$, the answer is "Yes."

The number of polynomials f to be examined is at most $n2^{n-1}$. For each polynomial f, we examine each segment C_k of C. If we obtain "Yes" for Problem 1 during the examination, the rest of the procedure is not needed. If we do not obtain "Yes" after the whole examination is done, then the answer is "No."

Remark 5. When $e_\lambda(x)$ is constant, $v(s, t)$ in (6) is a rational function only in s. In this case, when computing P and Q, we put $t = T(s)$ by solving t from $u = 0$ and set the left-hand side of (7) to v.

The order of the computational steps in the determination of whether f has a zero in D for a given polynomial $f \in F$ and whether α is a pseudozero of F for a given point $\alpha \in C$ is a polynomial in n. On the other hand, the order of the computational steps in the determination of whether there exists a pseudozero of F on C is 2^n times a polynomial in n and K, since the number of polynomials to be examined is of order 2^n as described above and the number of C_k is K. Therefore, the order of the total computational steps in the determination of whether there exists a pseudozero of F in D is 2^n times a polynomial in n and K.

4.4.3. Experiments. We carried out experimental computations for the following examples. We used the experimental computer algebra system Risa/Asir [10] on a computer with an Intel (R) Xeon$^{\text{TM}}$ processor (3.2 GHz) and 4 GB of memory.

Example 3. Solve Problem 1 for the interval polynomial

$$F = [0.9995, 1.0005]x^2 + [-0.6185, -0.6175]x + [0.9995, 1.0005]$$

and the domain $D = \{\, z \in \mathbb{C} \mid |z - (0.3096 + 0.9526 \cdot i)| \le 0.0004 \,\}$.

Example 4. Solve Problem 1 for the interval polynomial F in Example 3 and the domain D that is the rectangle whose vertexes are $0.3092 + i \cdot 0.95$, $0.31 + i \cdot 0.95$, $0.31 + i \cdot 0.953$ and $0.3092 + i \cdot 0.953$.

We obtained "No" for Example 3 and "Yes" for Example 4 within 0.1 s in both cases.

4.5. Rough Shape of Pseudozeros

Since, unlike real pseudozeros, the shape of pseudozeros is complicated, we should be content with a rough shape that is almost equal to the exact shape in general. For example, for an interval polynomial F, we should be content with a set $X \supset Z(F)$ that is a union of congruent closed rectangles intersecting $Z(F)$. We say that a set X is a rough shape for $Z(F)$ with precision ε, which is a positive real number, if the longest edge of the congruent rectangles is less than or equal to ε.

If the set $Z(F)$ is bounded and an initial rectangle containing $Z(F)$ is given, we can obtain a rough shape with arbitrary precision using the above computation methods for Theorem 3. If all polynomials in F have the same degree, we can compute an initial rectangle using, for example, the Cauchy bound for an algebraic equation. Once the initial rectangle is obtained, we divide it into four congruent rectangles and examine whether each of them intersects $Z(F)$. Similar computations are performed recursively for the rectangles that intersect $Z(F)$.

Note that we cannot use the Edge Theorem for determining a rough shape of pseudozeros of an interval polynomial since the outside of a rectangle does not satisfy the precondition for the Edge Theorem.

For efficient computation, several techniques are needed in order to avoid redundant computations. These remain for future study.

5. Conclusion

We have proposed a method for determining whether there exist a polynomial in a given interval polynomial that has a zero in a prescribed domain. The method is rigorous but is not efficient for a complex domain. Avoiding redundant computations, especially when computing a rough shape of pseudozeros, is one of our future directions. Another direction is to consider the following type of problem: For a given interval polynomial F and a given domain D, does every polynomial in F have a zero in D?

References

[1] A.C. Bartlett, C.V. Hollot and H. Lin, *Root location of an entire polytope of polynomials: it suffices to check the edges*, Mathematics of Controls, Signals and Systems, Vol. 1, pp. 61–71, 1988.

[2] R.M. Corless, H. Kai and S.M. Watt, *Approximate computation of pseudovarieties*, ACM SIGSAM Bulletin, Vol. 37, No. 3, pp. 67–71, 2003.

[3] M.A. Hitz and E. Kaltofen, *The Kharitonov theorem and its applications in symbolic mathematical computation*, Proc. Workshop on Symbolic-Numeric Algebra for Polynomials (SNAP96), pp. 20–21, 1996.

[4] M.A. Hitz and E. Kaltofen, *Efficient algorithms for computing the nearest polynomial with constrained roots*, Proc. 1998 International Symposium on Symbolic and Algebraic Computation (ISSAC98), pp. 236–243, 1998.

[5] M.A. Hitz, E. Kaltofen and Y.N. Lakshman, *Efficient algorithms for computing the nearest polynomial with a real root and related problems*, Proc. 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC99), pp. 205–212, 1999.

[6] J.W. Hoffman, J.J. Madden and H. Zhang, *Pseudozeros of multivariate polynomials*, Math. Comp., Vol. 72, No. 242, pp. 975–1002, 2003.

[7] E. Kaltofen, *Efficient algorithms for computing the nearest polynomial with parametrically constrained roots and factors*, Lecture at the Workshop on Symbolic and Numerical Scientific Computation (SNSC'99), 1999.

[8] V.L. Kharitonov, *Asymptotic stability of an equilibrium position of a family of systems of linear differential equations*, Differentsial'nye Uravneniya, Vol. 14, No. 11, pp. 2086–2088, 1978.

[9] R.G. Mosier, *Root neighborhoods of a polynomial*, Math. Comp., Vol. 47, No. 175, pp. 265–273, 1986.

[10] M. Noro and T. Takeshima, *Risa/Asir—A computer algebra system*, Proc. 1992 International Symposium on Symbolic and Algebraic Computation (ISSAC92), pp. 387–396, 1992.

[11] K. Shirayanagi, *An algorithm to compute floating point Gröbner bases*, Mathematical Computation with Maple V: Ideas and Applications, T. Lee (ed.), pp. 95–106, Birkhäuser, Boston, 1993.

[12] K. Shirayanagi, *Floating point Gröbner bases*, Mathematics and Computers in Simulation, Vol. 42, pp. 509–528, 1996.

[13] K. Shirayanagi and M. Sweedler, *A theory of stabilizing algebraic algorithms*, Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995.

[14] H.J. Stetter, *The nearest polynomial with a given zero, and similar problems*, ACM SIGSAM Bulletin, Vol. 33, No. 4, pp. 2–4, 1999.

Hiroshi Sekigawa and Kiyoshi Shirayanagi[1]
NTT Communication Science Laboratories
Nippon Telegraph and Telephone Corporation
3-1, Morinosato Wakamiya, Atsugi-shi,
Kanagawa, 243-0198 Japan
e-mail: `sekigawa@theory.brl.ntt.co.jp`
`shirayan@theory.brl.ntt.co.jp`[2]

[1]The current address is: Department of Mathematical Sciences, Tokai University, 1117 Kitakaname, Hiratsuka-shi, Kanagawa, 259-1292 Japan
[2]The current mail address is: `shirayan@ss.u-tokai.ac.jp`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 185–210
© 2007 Birkhäuser Verlag Basel/Switzerland

Root-Finding with Eigen-Solving

Victor Y. Pan, Dmitriy Ivolgin, Brian Murphy, Rhys Eric Rosholt, Yuqing Tang, Xinmao Wang and Xiaodong Yan

Abstract. We survey and extend the recent progress in polynomial root-finding via eigen-solving for highly structured generalized companion matrices. We cover the selection of eigen-solvers and matrices and show the benefits of exploiting matrix structure. No good estimates for the rate of global convergence of the eigen-solvers are known, but according to ample empirical evidence it is sufficient to use a constant number of iteration steps per eigenvalue. If so, the resulting root-finders are optimal up to a constant factor because they use linear arithmetic time per step and perform with a constant (double) precision. Some by-products of our study are of independent interest. The algorithms can be extended to solving secular equations.

Mathematics Subject Classification (2000). Primary 65H05; Secondary 65H17; Tertiary 65F15.

Keywords. Polynomial root-finding, eigenvalue, generalized companion matrix, secular equation.

1. Introduction

1.1. Background

Polynomial root-finding is a classical and highly developed area but is still an area of active research [McN93, McN97, McN99, McN02, NAG88, P97, P01/02, PMRTa]. The divide-and-conquer algorithms in [P95, P96, P01/02] (cf. [S82], [G52/58, CN94, NR94, K98] on some important related works) approximate all roots of a polynomial by using arithmetic and Boolean time which is optimal up to polylogarithmic factors (under both sequential and parallel models of computing). The algorithm, however, is quite involved, and the users prefer more transparent iterative algorithms, such as Newton's, Jenkins-Traub's [JT70, JT72], Müller's, Laguerre's, and Halley's, which use linear arithmetic time per iteration and approximate a single root, and Durand-Kerner's (actually Weierstrass') and

Supported by PSC CUNY Awards 66437-0035 and 67297-0036.

Aberth/Ehrlich's (actually Börsch-Supan's), which use quadratic time per iteration and approximate all roots of a polynomial (see Tables 4–6). The iterations converge superlinearly if the approximations are close to the roots.

Computing close initial approximations is still an unsettled area. A popular approach is to seek them as approximations to the eigenvalues of the Frobenius companion matrix, whose spectrum is precisely the set of the roots of the polynomial. This property characterizes the more general class of generalized companion (hereafter we say *GC*) matrices of a polynomial, which can be used instead of the Frobenius matrix and, like it, can be chosen highly structured. Thus one can first approximate the eigenvalues of a GC matrix numerically, by exploiting its structure and employing the highly effective software of numerical eigen-solvers, and then refine the approximations rapidly, by applying the cited polynomial root-finders. Such a combination of the power of numerical techniques of structured matrix computations and symbolic/algebraic methods of computations with polynomials naturally continues the extensive study in [P92, BP94, BP94, P98, P98/01, MP00, P01, EP02, BGP02/04, EMP04, BGP03/05] and the references therein. We contribute to this area once again, although we only cover eigen-solving, not the refining stage.

1.2. The QR DPR1 Approach

Matlab approximates polynomial roots by applying the QR eigen-solver to the Frobenius matrix. This works quite well except that the output approximations to the eigenvalues are frequently too crude and need refinement.

Malek and Vaillantcourt in [MV95, MV95a] and Fortune in [F01/02] apply the QR algorithm to the diagonal plus rank-one (hereafter we say *DPR1*) GC matrices, defined by the polynomial and the root approximations, which we call the *companion knots*. As soon as the QR algorithm stops and outputs the updated knots, the matrix is updated as well, and the QR algorithm is reapplied to it. According to the extensive tests reported in the three papers and some theory in [F01/02], this process indeed improves the approximations rapidly until they initialize the cited popular root-finders.

In [BGP03/05, BGP04] the rank structure of the DPR1 input matrix has been exploited to accelerate the QR stage of the algorithms in [MV95, MV95a, F01/02] by the order of magnitude. The resulting algorithm uses linear (rather than quadratic) memory space and linear arithmetic time per iteration, but otherwise performs as the classical QR algorithm, remaining as robust and converging as rapidly. The acceleration, however, is achieved only where the companion knots are real or, with the amendment in [BGP04] based on the Möbius transform of the complex plane, where they lie on a line or circle. Thus the algorithms in [BGP03/05, BGP04] use linear space and linear time per step only for the original DPR1 matrix, but not for its updates.

1.3. Improved DPR1 Eigen-Solving

To fix this deficiency we employ rather simple means. We examine other polynomial root-finders and matrix eigen-solvers in lieu of or in addition to the algorithms used in [MV95, MV95a, F01/02], and we propose an alternating application of various algorithms in a unified recursive process for root-finding. In [BGP02/04] the inverse power iteration with Rayleigh quotients (hereafter we say the *IPI*) is applied to the Frobenius and DPR1 GC matrices. It is immediately verified that in the case of a DPR1 input, linear memory space and linear arithmetic time are sufficient per an IPI step (as well as for the QR step in [BGP03/05, BGP04]) and also for deflating a DPR1 matrix. The algorithm in [BGP02/04] is initialized with the companion knots on a large circle, which is a customary recipe for polynomial root-finding. The IPI, however, converges faster near an eigenvalue. This motivates using a hybrid algorithm where the IPI refines the crude approximations computed by the QR algorithm.

For the IPI, QR, and all other popular eigen-solvers no good upper bounds are known on the number of steps they need for convergence. According to the ample empirical evidence, however, a single QR step as well as a single step of the IPI (initialized near the solution) is typicaly sufficient per an eigenvalue [GL96, pages 359 and 363]. (See our Sect. 6.1 or [P05] on a nontrivial technique of convergence acceleration for the IPI.) Under the latter semi-empirical model, the hybrid polynomial root-finders based on eigen-solving perform $O(n^2)$ ops with the double precision of d bits, that is, $O((n^2 d \log d) \log\log d)$ bit-operations, to approximate all roots sufficiently closely to initialize the Newton's or Weierstrass' refinement.

This cost is within the factor of $(\log d) \log\log d$ from an information lower bound. (The factor is a constant if so is d.) Indeed, one needs at least n complex numbers to represent the coefficients of a monic input polynomial $c(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$, and needs at least the order of $(n-i)d$ bits in each coefficient c_i to approximate the roots within the error $2^{-d} \max_j |c_j|$. This means the order of $n^2 d$ bits in all coefficients. Therefore, at least the same order of Boolean operations is required to process these bits.

Unlike the nearly optimal algorithm in [P01/02], the eigen-solving approach has the more limited goal of obtaining close initial approximations for polynomial root-finders and requires no computations with the extended precision.

1.4. Extensions and Further Study

How much can the progress be pushed further? According to the above argument, at most by a constant factor. This can still be practically important. The natural avenues are by exploting effective eigen-solvers such as Arnoldi's, non-Hermitian Lanczos', and Jacobi-Davidson's (besides the QR and IPI), applying them to the DPR1, Frobenius and other relevant GC matrices, and combining these eigen-solvers with some popular polynomial root-finders. We estimate the computational time for multiplication of these GC matrices and their shifted inverses by vectors, for deflation, and for updating a GC matrix when its companion knot changes.

Besides, we observe that the eigen-solvers also support some new proximity tests for the roots as well as the computation of the basic root-free annuli for polynomial factorization (see Sect. 6.5). We also comment on the extension of the algorithms to approximating the eigenvalues of sparse and structured matrices and to solving secular equations. On further applications of these equations, see [G73, M97, BP98] and the references therein.

For simplicity we narrow our study to monic input polynomials and skip the important special case of polynomials that have only real roots. See [JV04, BP98] and the bibliography therein on these omitted subjects.

1.5. Organization of Our Paper

We organize our paper as follows. In Sect. 2, we recall some basic definitions. In Sect. 3, we study some relevant classes of GC matrices. In Sect. 4, we estimate the arithmetic computational complexity of some basic operations with these matrices. In Sect. 5, we study their computation, deflation, and updating. In Sect. 6, we cover various aspects of the application of eigen-solving for these matrices to polynomial root-finding. In Sect. 7, we comment on the extension of our methods to approximating matrix eigenvalues. In Sect. 8, we recall the correlation between the polynomial and secular equations. In the Appendix, we comment on heuristics for multiple roots and root clusters and on computing approximate polynomial gcds. All authors share the responsibility for extensive numerical tests that supported the presented exposition and analysis. Otherwise the paper is due to the first author.

2. Basic Definitions

$M = (m_{i,j})_{i,j=1}^n$ is an $n \times n$ matrix, $\mathbf{v} = (v_i)_{i=1}^n$ is a column vector of dimension n, M^T and $\mathbf{v}^T$ are their transposes.

$0_{k,l}$ is the $k \times l$ null matrix, $\mathbf{0}_k = 0_{k,k}$. I_k is the $k \times k$ identity matrix. I is the identity matrix of an appropriate size. $\mathbf{e}_i$ is the i-th column of I_n, $i = 1, \ldots, n$; $\mathbf{e}_1 = (1, 0, \ldots, 0)^T$, $\mathbf{e}_n = (0, \ldots, 0, 1)^T$.

$B = (B_1, \ldots, B_k)$ is the $1 \times k$ block matrix with blocks $B_1, \ldots, B_k$. $\mathrm{diag}(s_i)_{i=1}^n$ is the $n \times n$ diagonal matrix with the diagonal entries $s_1, \ldots, s_n$. $\mathrm{diag}(B_1, \ldots, B_k)$ is the $k \times k$ block diagonal matrix with the diagonal blocks $B_1, \ldots, B_k$.

$\det M$ and $c_M(\lambda) = \det(\lambda I - M)$ are the determinant and the characteristic polynomial of a matrix M, respectively.

$$Z = (z_{i,j})_{i,j=1}^n = \begin{pmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}$$ is the $n \times n$ shift matrix, $z_{i,i-1} = 1$ for $i = 2, \ldots, n$; $z_{i,j} = 0$ for $i \neq j + 1$, $Z\mathbf{v} = (0, v_1, \ldots, v_{n-1})^T$ for $\mathbf{v} = (v_i)_{i=1}^n$. Here

and hereafter the blank space in the representation of matrices stands for their zero entries.

$f^* = a - b\sqrt{-1}$ is the complex conjugate of $f = a + b\sqrt{-1}$, for real $a = \Re f$ and $b = \Im f$. $\omega_n = \exp(2\pi\sqrt{-1}/n)$ is a primitive n-th root of 1.

$$V = \frac{1}{\sqrt{n}}(\omega_n^{ij})_{i,j=0}^{n-1} \tag{2.1}$$

is the unitary matrix of the discrete Fourier transform on the n-th roots of 1.

"DPR1", "GC", "IPI", "RBDPR1", and "TPR1" stand for "diagonal plus rank-one", "generalized companion", "Inverse Power Iteration", "real block diagonal plus rank-one", and "triangular plus rank-one", respectively. In Sects. 3 and 7, "ops" stands for "arithmetic operations". In the Appendix, "gcd" stands for "greatest common divisor".

$C = C_c$ is a GC matrix for a monic polynomial

$$c(x) = c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0,\ c_n = 1, \tag{2.2}$$

if $c_C(x) = c(x)$.

3. Some Classes of GC Matrices

Root-finding for a polynomial $c(x)$ in (2.2) is equivalent to eigen-solving for a GC matrix $C = C_c$. The efficiency of the eigen-solving greatly depends on the choice of the matrix. Next we examine some most relevant classes of GC matrices (compare the studies of GC matrices in [E73, G73, B75, F90, C91, MV95]).

3.1. The Frobenius Companion Matrix

We first recall the classical Frobenius companion martix.

Theorem 3.1. *The $n \times n$ matrix*

$$C = F_c = \begin{pmatrix} 0 & & & & -c_0 \\ 1 & \ddots & & & -c_1 \\ & \ddots & \ddots & & \vdots \\ & & \ddots & 0 & -c_{n-2} \\ & & & 1 & -c_{n-1} \end{pmatrix} \tag{3.1}$$

is a GC matrix F_c for a monic polynomial $c(x)$ in (2.2).

$C = F_c = Z - \mathbf{c}\mathbf{e}_n^T$ for $\mathbf{c} = (c_i)_{i=0}^{n-1}$.

3.2. DPR1 GC Matrices

Theorem 3.2. *For a polynomial $c(x)$ in (2.2) and n distinct scalar companion knots $s_1, \dots, s_n$, write*

$$\mathbf{s} = (s_i)_{i=1}^n, q(x) = \prod_{i=1}^n (x-s_i), q_i(x) = \prod_{j=1, j\neq i}^n (x-s_j) = \frac{q(x)}{x-s_i},\ i = 1, \dots, n, \quad (3.2)$$

$$d_i = \frac{c(s_i)}{q'(s_i)},\ i = 1, \dots, n, \quad (3.3)$$

$$\mathbf{u} = (u_i)_{i=1}^n,\ \mathbf{v} = (v_i)_{i=1}^n,\ B = B_{\mathbf{s}} = \operatorname{diag}(s_i)_{i=1}^n,\ C = B - \mathbf{u}\mathbf{v}^T \quad (3.4)$$

where

$$|u_i| + |v_i| \neq 0,\ d_i = u_i v_i,\ i = 1, \dots, n. \quad (3.5)$$

Then C is a DPR1 GC matrix for the polynomial $c(x)$, that is, $c_C(x) = c(x)$.

Proof. $c_C(x) = q(x) + \sum_{i=1}^n d_i q_i(x)$ because the i-th and j-th rows of the matrix $xI - C - \operatorname{diag}(0, x - s_i, 0) - \operatorname{diag}(0, x - s_j, 0)$ for $i \neq j$ are proportional to one another, whereas $c(x) = q(x) + \sum_{i=1}^n d_i q_i(x)$ due to the Lagrange interpolation formula. $\square$

3.3. RBDPR1 GC Matrices

The polynomials $c(x)$ in (2.2) with real coefficients may have some pairs of nonreal complex conjugate roots. In this case the DPR1 matrices would have nonreal entries. To avoid this deficiency we introduce the Real Block DPR1 (hereafter we say *RBDPR1*) GC matrices whose diagonal blocks have size of at most two. We begin with an auxiliary result on block diagonal plus rank-one matrices.

Theorem 3.3. *Let $B = \operatorname{diag}(B_1, \dots, B_k)$ where B_i are $n(i) \times n(i)$ matrices, $m(i) = \sum_{j=1}^i n(j)$, $i = 1, \dots, k$, $m(k) = n$. Write*

$$P_i = \operatorname{diag}(0_{m(i-1)}, I_{n(i)}, 0_{n-m(i)}), \overline{P}_i = (0_{n(i),m(i-1)}, I_{n(i)}, 0_{n(i),n-m(i)}),$$

so that $\overline{P}_i \mathbf{w} = (w_j)_{j=m(i-1)+1}^{m(i)}$ is the projection of a vector $\mathbf{w} = (w_j)_{j=1}^n$ into its subvector made up of the $n(i)$ respective coordinates, whereas by padding the vector $\overline{P}_i \mathbf{w}$ with the $m(i-1)$ leading zero coordinates and the $n - m(i)$ trailing zero coordinates, we arrive at the vector $P_i \mathbf{w}$. Let s_i be an eigenvalue of the matrix B and let $C = B - \mathbf{u}\mathbf{v}^T$. Then $c_C(s_i) = \det(s_i I - B + P_i \mathbf{u}\mathbf{v}^T P_i) = \det(s_i I - B_i + \overline{P}_i \mathbf{u}\mathbf{v}^T \overline{P}_i) \prod_{j \neq i} \det(s_i I - B_j) = c_B(s_i), i = 1, \dots, n$.

Proof. Let $\mathbf{q}_i \neq \mathbf{0}$ be a left eigenvector of the matrix B associated with the eigenvalue s_i such that $B\mathbf{q}_i^T = s_i \mathbf{q}_i$. Write $a_i = \mathbf{q}_i^T \mathbf{u}$ and $\mathbf{u} = (u_j)_{j=1}^n$. If $a_i = 0$, then we have $\mathbf{q}_i^T(s_i I - B) = \mathbf{q}_i^T(s_i I - C) = \mathbf{0}^T$ and therefore $c_B(s_i) = c_C(s_i) = 0$. Otherwise subtract the vector $\frac{u_j}{a_i} \mathbf{q}_i^T(s_i I - B + \mathbf{u}\mathbf{v}^T) = \frac{u_j}{a_i} \mathbf{q}_i^T \mathbf{u}\mathbf{v}^T = u_j \mathbf{v}^T$ from the j-th row of the matrix $s_i I - C = s_i I - B + \mathbf{u}\mathbf{v}^T$ for $j = 1, \dots, m(i-1)$, and for $j = m(i)+1, \dots, n$. This turns the matrix $s_i I - C$ into the matrix $s_i I - B + P_i \mathbf{u}\mathbf{v}^T$ without changing its determinant $c_C(s_i)$. Observe that $\det(s_i I - B + P_i \mathbf{u}\mathbf{v}^T) =$

$\det(s_iI - B + P_i\mathbf{u}\mathbf{v}^TP_i)$ and that $s_iI - B + P_i\mathbf{u}\mathbf{v}^TP_i$ is a block diagonal matrix with the diagonal blocks $s_jI_{n(j)} - B_j$ for $j = 1,\dots,i-1,i+1,\dots,k$ and $s_iI_{n(i)} - B_i + \overline{P}_i\mathbf{u}\mathbf{v}^T\overline{P}_i^T$. This proves Theorem 3.3. □

Theorem 3.3 enables the following alternative proof of Theorem 3.2.

Proof. (*An alternative proof of Theorem* 3.2.) Apply Theorem 3.3 for $B_j = (s_j)$, $j = 1,\dots,n$, $B = \mathrm{diag}(s_j)_{j=1}^n$, $k = n$, $n_i = 1$, $i = 1,\dots,n$. Obtain that $c_C(s_i) = d_iq_i(s_i)$, substitute $q_i(s_i) = q'(s_i)$, $d_iq'(s_i) = c(s_i)$, and obtain that $c(s_i) = c_C(s_i)$, $i = 1,\dots,n$. This proves the theorem because $c(x)$ and $c_C(x)$ are monic polynomials of degree n. □

Theorem 3.4. *For two integers h and n, $0 \le h \le \frac{n}{2}$, a polynomial $c(x)$ in (2.2) with real coefficients, h distinct pairs of real numbers $(f_1, g_1),\dots,(f_h, g_h)$ such that $g_i \ne 0$ for all i, and $n-2h$ distinct real numbers $s_{2h+1},\dots,s_n$, write $s_{2i-1} = f_i + g_i\sqrt{-1}$, $s_{2i} = f_i - g_i\sqrt{-1}$, $B_i = \begin{pmatrix} f_i & g_i \\ -g_i & f_i \end{pmatrix}$, $i = 1,\dots,h$; $B_{j-h} = (s_j)$, $j = 2h+1,\dots,n$, $B = \mathrm{diag}(B_j)_{j=1}^{n-h}$; $q(x) = \prod_{j=1}^n(x - s_j)$, $d_j = \frac{c(s_j)}{q'(s_j)}$, $j = 1,\dots,n$, so that $d_{2i} = d_{2i-1}^*$, $i = 1,\dots,h$. Let*

$$\mathbf{u} = (u_j)_{j=1}^n,\ \mathbf{v} = (v_j)_{j=1}^n,\ C = B - \mathbf{u}\mathbf{v}^T \tag{3.6}$$

where

$$u_{2i-1}v_{2i-1} + u_{2i}v_{2i} + (u_{2i-1}v_{2i} - u_{2i}v_{2i-1})\sqrt{-1} = 2d_{2i-1},$$

for $i = 1,\dots,h$, $|u_j| + |v_j| \ne 0$, $u_jv_j = d_j$, $j = 2h+1,\dots,n$.

Then the RBDPR1 matrix C is a GC matrix of the polynomial $c(x)$, that is, $c(x) = c_C(x)$.

Proof. Apply Theorem 3.3 for $k = n-h$ and deduce that $(s_{2i-1} - s_{2i})c_C(s_{2i-1}) = q_{2i-1}(s_{2i-1})\det(s_{2i-1}I_2 - W_i)$ for

$$W_i = B_i - \overline{P}_i\mathbf{u}\mathbf{v}^T\overline{P}_i = \begin{pmatrix} f_i - u_{2i-1}v_{2i-1} & g_i - u_{2i-1}v_{2i} \\ -g_i - u_{2i}v_{2i-1} & f_i - u_{2i}v_{2i} \end{pmatrix}, i = 1,\dots,h.$$

Substitute $s_{2i-1} - f_i = g_i\sqrt{-1}$ and deduce that

$$s_{2i-1}I_2 - W_i = \begin{pmatrix} g_i\sqrt{-1} + u_{2i-1}v_{2i-1} & -g_i + u_{2i-1}v_{2i} \\ g_i + u_{2i}v_{2i-1} & g_i\sqrt{-1} + u_{2i}v_{2i} \end{pmatrix},$$

so that $\det(s_{2i-1}I_2 - W_i) = g_i(u_{2i}v_{2i-1} - u_{2i-1}v_{2i} + (u_{2i-1}v_{2i-1} + u_{2i}v_{2i})\sqrt{-1})$, $i = 1, \dots, h$. Substitute the latter expression and the equations $s_{2i-1} - s_{2i} = 2g_i\sqrt{-1}$ and $q_j(s_j) = q'(s_j)$ for $j = 2i-1$ into our expression above for $c_C(s_{2i-1})$ and obtain that

$$2g_ic_C(s_{2i-1})\sqrt{-1} = g_iq'(s_{2i-1})((u_{2i-1}v_{2i-1} + u_{2i}v_{2i})\sqrt{-1} + u_{2i}v_{2i-1} - u_{2i-1}v_{2i}),$$

$$u_{2i-1}v_{2i-1} + u_{2i}v_{2i} + (u_{2i-1}v_{2i} - u_{2i}v_{2i-1})\sqrt{-1} = \frac{2c_C(s_{2i-1})}{q'(s_{2i-1})} = 2d_{2i-1}.$$

Now apply equation (3.3) and deduce that $c_C(s_{2i-1}) = c(s_{2i-1})$ for $i = 1, \dots, h$.

Since the polynomials $c(x)$ and $c_C(x)$ have real coefficients, obtain that $c_C(s_{2i}) = c_C^*(s_{2i-1}) = c^*(s_{2i-1}) = c(s_{2i})$, $i = 1,\dots,h$. Deduce that $c_C(s_j) = c(s_j)$

by applying Theorem 3.3 for $B_j = \mathrm{diag}(s_j), j = 2h+1, \ldots, n$. Now Theorem 3.4 follows because $c(x)$ and $c_C(x)$ are monic polynomials of degree n. □

3.4. Arrow-Head GC Matrices

Theorem 3.5. *For a polynomial $c(x)$ in (2.2) and n distinct nonzero scalars $\overline{s}_1, \ldots, \overline{s}_n$, write*

$$\overline{q}(x) = \prod_{i=2}^{n}(x - \overline{s}_i),\ \overline{q}_i(x) = \prod_{j=2, j\neq i}^{n}(x - \overline{s}_j) = \frac{\overline{q}(x)}{x - \overline{s}_i},\ i = 2, \ldots, n, \tag{3.7}$$

$$\overline{d}_i = \frac{c(\overline{s}_i)}{\overline{q}'(\overline{s}_i)} = \frac{c(\overline{s}_i)}{\overline{q}_i(\overline{s}_i)},\ i = 2, \ldots, n,\ \overline{d}_1 = \frac{c(\overline{s}_1)}{\overline{q}(\overline{s}_1)} + \sum_{i=2}^{n} \frac{\overline{d}_i}{\overline{s}_1 - \overline{s}_i} \tag{3.8}$$

and choose n pairs of scalars $\overline{u}_i$, $\overline{v}_i$, $i = 1, \ldots, n$ such that

$$\overline{u}_1 = \overline{d}_1 - \overline{s}_1,\ \overline{v}_1 = 0,\ \overline{u}_i\overline{v}_i = \overline{d}_i,\ i = 2, \ldots, n. \tag{3.9}$$

Write $B = B_{\overline{s}} = \mathrm{diag}(\overline{s}_i)_{i=1}^n$, $\overline{\mathbf{u}} = (\overline{u}_i)_{i=1}^n, \overline{\mathbf{v}} = (\overline{v}_i)_{i=1}^n$. Then the north-western arrow-head matrix

$$C = B - (\overline{\mathbf{u}}\mathbf{e}_1^T + \mathbf{e}_1\overline{\mathbf{v}}^T) \tag{3.10}$$

is a GC matrix of the polynomial $c(x)$, that is, $c_C(x) = c(x)$.

Proof. Expand the determinant $c_C(x) = \det(xI - C)$ along the first row or the first column of the matrix $xI - C$ and deduce that

$$c_C(x) = (x + \overline{u}_1)\overline{q}(x) - \sum_{i=2}^{n} \overline{u}_i\overline{v}_i\overline{q}_i(x).$$

Therefore,

$$c_C(\overline{s}_i) = \overline{u}_i\overline{v}_i\overline{q}_i(\overline{s}_i),\ i = 2, \ldots, n;$$
$$c_C(\overline{s}_1) = (\overline{s}_1 + \overline{u}_1)\overline{q}(\overline{s}_1) - \sum_{i=2}^{n} \overline{u}_i\overline{v}_i\overline{q}_i(\overline{s}_1).$$

Substitute equations (3.8) and (3.9) and deduce that $c_C(\overline{s}_i) = c(\overline{s}_i), i = 1, \ldots, n$. The theorem follows because the monic polynomials $c_C(x)$ and $c(x)$ of degree n share their values at n distinct points $\overline{s}_1, \ldots, \overline{s}_n$. □

3.5. Further Variations of GC Matrices

The Frobenius, DPR1, and arrow-head matrices are the most popular classes of GC matrices. The RBDPR1 GC matrices extend the DPR1 GC matrices in the case of a real input and a nonreal output. Similarly we can extend the class of the arrow-head matrices. Let us point out some further variations and extensions.

1. *Variations of the parameters.*

   For fixed companion knots, each GC matrix in Sects. 3.2–3.4 is defined with n or $n-1$ parameters, which we can vary at will.

Example 3.1. *Some sample choices of the parameters.*

- $u_i = 1,\ v_i = d_i,\ i = 1, \ldots, n,$ *in Theorem* 3.2

- $v_{2i-1} = v_{2i} = 1,\ u_{2i-1} = \Re d_i + \Im d_i,\ u_{2i} = \Re d_i - \Im d_i,\ i = 1, \dots, h,$ $u_j = 1,\ v_j = d_j,\ j = 2h+1, \dots, n,$ *in Theorem* 3.4
- $\overline{u}_i = 1,\ \overline{v}_i = \overline{d}_i,\ i = 2, \dots, n,$ *in Theorem* 3.5

Example 3.2. *Scaling for numerical stabilization.*
In Theorem 3.2 *require that* $|u_i| = |v_i|$ *(resp.* $|\overline{u}_i| = |\overline{v}_i|$*) for all* i*.*

2. *Variation of the input polynomial.*
   We can fix a scalar b and then apply Theorem 3.5 to the polynomial $(x-b)c(x)$ with a root b to approximate the remaining n roots. Applying the theorem, we replace $c(x)$ with $(x-b)c(x)$, replace n with $n+1$, and choose $s_1 = b$, so that $\overline{d}_1 = \sum_{i=2}^{n} \frac{\overline{d}_i}{\overline{s}_1 - \overline{s}_i}$.
3. *Modification of the matrices.*
   - We can extend Theorem 3.4 by choosing any set of $2h$ real 2×2 matrices $B_i = \begin{pmatrix} f_i & g_i \\ j_i & k_i \end{pmatrix},\ i = 1, \dots, h$ and any set of $n - 2h$ real 1×1 matrices $B_j = (s_j),\ j = 2i+1, \dots, n$, with n distinct eigenvalues overall. Suppose s_{2i-1} and s_{2i} denote the eigenvalues of the matrix $B_i,\ i = 1, \dots, h$. Then for any choice of the values $u_{2i-1}, u_{2i}, v_{2i-1}, v_{2i}$ satisfying

$$(s_{2i-1} - f_i)u_{2i}v_{2i} + (s_{2i-1} - j_i)u_{2i-1}v_{2i-1} + g_i u_{2i}v_{2i-1} + h_i u_{2i-1}v_{2i}$$
$$= 2(s_{2i-1} - s_{2i})d_{2i-1},\ i = 1, \dots, h,$$

     the matrix C in (3.6) is a GC matrix of the polynomial $c(x)$.
   - We can interchange the roles of the subscripts 1 and n throughout Theorem 3.5 to arrive at the dual south-eastern arrow-head matrix C such that $c_C(x) = c(x)$. Alternatively, we can turn a north-western arrow-head matrix into a south-eastern one by applying the similarity transform $C \longrightarrow JCJ$ where $J = J^{-1}$ is the reflection matrix whose entries equal one on the antidiagonal and equal zero elsewhere.
   - More generally, any similarity transform $C \longrightarrow S^{-1}CS$ of a GC matrix $C = C_c$ for a polynomial $c(x)$ maps C into a GC matrix for $c(x)$. If all n roots of $c(x)$ are distinct, then the converse is also true, that is, two GC matrices associated with such a polynomial $c(x)$ are always similar to one another. In the next subsection we specify such transforms among our sample GC matrices. The similarity transforms can be of some help in actual computations, e.g., with appropriate diagonal matrices S we can scale the GC matrices to improve their conditioning. This diagonal scaling of GC matrices is equivalent to choosing n parameters among $u_i, v_i,\ i = 1, \dots, n$ in Sects. 3.2 and 3.3 or $n - 1$ parameters among $\overline{u}_i, \overline{v}_i,\ i = 1, \dots, n$ in Sect. 3.4.

3.6. Similarity Transforms Among GC Matrices of Four Classes

Simple similarity transforms of a 2×2 matrix $B = \begin{pmatrix} f_i & g_i \\ -g_i & f_i \end{pmatrix}$ into the diagonal matrix $\mathrm{diag}(d_{2i-1}, d_{2i})$, $d_{2i-1} = f_i + g_i\sqrt{-1}$, $d_{2i} = f_i - g_i\sqrt{-1}$ can be immediately extended to transforming a block diagonal matrix B in Theorem 3.3 into a diagonal matrix. This relates the matrix classes DPR1 and RBDPR1 in Sects. 3.2 and 3.3 and similarly for the arrow-head matrices in Sect. 3.4 and their counter-parts where the diagonal entries can be replaced by real blocks.

Furthermore, both arrow-head matrix C in (3.10) and the transpose F_c^T of a Frobenius matrix F_c in (3.1) are TPR1 matrices, and the paper [PMRTYCa] shows non-unitary similarity transforms of TPR1 into DPR1 matrices as well as into arrow-head matrices. For the matrices F_c^T and C in (3.10), these transforms into DPR1 matrices use $O(n^2)$ ops. There are also similarity transforms of our matrices in (3.4), (3.6) and (3.10) into a Frobenius matrix via their reduction to a Hessenberg matrix in [W65, pages 405–408] as well as a unitary similarity transform of a matrix F_c into a DPR1 matrix due to the following result.

Theorem 3.6. *The similarity transform with the matrix V in* (2.1) *maps the Frobenius matrix F_C in* (3.1) *into a $DPR1$ matrix:*

$$VF_CV^H = \operatorname{diag}(w_n^i)_{i=0}^{n-1} + \mathbf{uv}^T, \ \mathbf{u} = V\mathbf{c}, \ \mathbf{v}^T = \mathbf{e}_n^T V^H.$$

4. The Complexity of Some Basic Computations

Multiplication of the input matrices and their shifted inverses with vectors are basic operations in some popular eigen-solvers. Tables 1 and 2 and Theorem 4.1 show arithmetic complexity of these operations for the matrices C in (3.1)–(3.10).

In the columns of Tables 1 and 2 marked by a/s, m, and r we show how many times we add/subtract, multiply, and compute reciprocals, respectively, to arrive at the vectors $C\mathbf{w}$, $(xI - C)^{-1}\mathbf{w}$, and $(xI - C - \mathbf{gh}^T)^{-1}\mathbf{w}$ for a fixed pair of vectors $\mathbf{g}$ and $\mathbf{h}$, any scalar x such that the matrices $xI - C$ and $xI - C - \mathbf{gh}^T$ are nonsingular, and any vector $\mathbf{w}$. Some entries of Table 2 have two levels. In the upper level the number of ops depending on the vector $\mathbf{w}$ is displayed; in the low level the number of the other ops is displayed. All estimates hold where the parameters u_i, v_i, $\overline{u}_i$, and $\overline{v}_i$ satisfy the equations in Example 3.1. For other choices of the parameters the arithmetic cost can slightly change.

Table 1. The complexity of multiplication of GC matrices by a vector

Matrix C	Vectors		$C\mathbf{w}$	
	$\mathbf{g}$	$\mathbf{h}$	m	a/s
Frobenius in (3.1)	$\mathbf{c}$	$\mathbf{e}_n$	n	$n-1$
DPR1 in (3.4)	$\mathbf{u}$	$\mathbf{v}$	$2n-1$	$2n-2$
RBDPR1 in (3.6)	$\mathbf{u}$	$\mathbf{v}$	$2n+2h$	$2n$
Arrow-head in (3.10)	$\mathbf{e}_1$	$-\mathbf{v}$	$2n-1$	$2n-2$

Theorem 4.1. *Let a polynomial $c(x)$ and scalars s_i, d_i, u_i, v_i, $\overline{s}_i$, $\overline{d}_i$, $\overline{u}_i$, and $\overline{v}_i$ for $i = 1, \ldots, n$, satisfy equations* (3.1)–(3.10). *Let four matrices, all denoted by C, satisfy equations* (3.1), (3.4), (3.6), *and* (3.10), *respectively, and let $\overline{C} = Z$ for*

TABLE 2. The complexity of multiplication of the shifted inverse matrices by a vector

Matrix C	Vectors		$(xI-C)^{-1}\mathbf{w}$			$(xI-\overline{C})^{-1}\mathbf{w}$		
	$\mathbf{g}$	$\mathbf{h}$	m	a/s	r	r	m	a/s
Frobenius	$\mathbf{c}$	$\mathbf{e}_n$	0	$2n-1$	$2n-2$	0	n	$n-1$
in (3.1)			1	$n-1$	n	1	0	1
DPR1	$\mathbf{u}$	$\mathbf{v}$	1	$2n$	$2n-1$	0	n	0
in (3.4)			$n+1$	$n+1$	$2n$	n	0	n
RBDPR1	$\mathbf{u}$	$\mathbf{v}$	$n+1$	$n+h$	$2n-1+2h$	n	h	$2h$
in (3.6)			$2n$	$2h$	$n+3h$	h	h	h
Arrow-head	$\mathbf{e}_1$	$-\mathbf{v}$	0	$2n-1$	$2n-2$	0	n	$n-1$
in (3.10)			n	$n-1$	$2n-1$	n	0	n

C in (3.1), $\overline{C}=B$ for C in (3.4) and (3.6), and $\overline{C}=B+\overline{\mathbf{u}}\mathbf{e}_1^T$ for C in (3.10), so that $C-\overline{C}$ denotes the rank-one matrices $-\mathbf{c}\mathbf{e}_n^T$, $-\mathbf{u}\mathbf{v}^T$, and $\mathbf{e}_1\overline{\mathbf{v}}^T$, respectively. Let x be a scalar such that the matrices $xI-C$ and $xI-\overline{C}$ are nonsingular. Let $\mathbf{w}$ be a vector. Then Tables 1 and 2 display the upper bounds on the numbers of the operations a/s, m, and r involved in computing the vectors $C\mathbf{w}$, $(xI-C)^{-1}\mathbf{w}$, and $(xI-\overline{C})^{-1}\mathbf{w}$. For the two latter vectors, an upper bound on the number of the ops not depending on the vector $\mathbf{w}$ is showed in the lower level of each entry of Table 2. The other ops are counted in its upper level.

Proof. The straightforward algorithms support the estimates for the complexity of computing the vectors $C\mathbf{w}$ and $(xI-\overline{C})^{-1}\mathbf{w}$. (Apply the forward substitution algorithm under (3.10) for $(xI-\overline{C})^{-1}\mathbf{w}$.)

Compute the vectors $(xI-C)^{-1}\mathbf{w}$ for the matrices C in (3.1) and (3.10) by applying Gaussian elimination. For a Frobenius matrix C in (3.1), first eliminate the subdiagonal entries by using no pivoting and then apply the back substitution. For an arrow-head matrix C in (3.10), first eliminate the first row of the matrix and then apply the forward substitution. Verify the respective estimates in Table 2 by inspection.

The Sherman-Morrison-Woodbury formula ([GL96, page 50] and [BGP02/04, Sect. 5]) implies that $(xI-C)^{-1}=(I+\frac{1}{1-\tau}(B-xI)^{-1}\mathbf{d}\mathbf{e}^T)(xI-B)^{-1}$, $\tau=\mathbf{e}^T(xI-B)^{-1}\mathbf{d}$, for $\mathbf{e}=(1,\dots,1)^T$ and the DPR1 matrix C in (3.4). Therefore, $(xI-C)^{-1}\mathbf{w}=(xI-B)^{-1}\mathbf{w}+\frac{\sigma}{1-\tau}(B-xI)^{-1}\mathbf{d}$, $\sigma=\mathbf{e}^T(B-xI)^{-1}\mathbf{w}$, and the estimates in Table 2 follow. Similarly cover the RBDPR1 matrices. $\square$

5. The Computation, Deflation, and Updating of a GC Matrix

This section covers the computation of a GC matrix, its deflation, and its updating when the companion knots and the input polynomial are modified.

5.1. The Computation of a GC Matrix

The matrix $C = F_c$ in (3.1) is given with the coefficients of the polynomial $c(x)$. The computation of the GC matrices C of the other three classes can be exemplified with the case of the DPR1 matrices in (3.4) and can be reduced essentially to computing the ratios $d_j = \frac{c(s_j)}{q'(s_j)}$ at the n distinct companion knots s_j, $j = 1, \ldots, n$.

The computation is simplified for the customary initial choice of the knots equally spaced on a large circle such that $s_j = a\omega_n^{j-1}$, $j = 1, \ldots, n$, where a exceeds by a sufficiently large factor the root radius $r = \max_j |z_j|$ of the polynomial $c(x) = \prod_{j=1}^{n}(x - z_j)$. In this case $q(x) = x^n - a^n$, $q'(x) = nx^{n-1}$. Then application of the generalized discrete Fourier transform [P01, Sect. 2.4] yields all ratios d_j in (3.3) by using $O(n \log n)$ ops. (Surely if n is a power of two, then one should just apply FFT.)

If, however, some crude initial approximations to the roots are available, they are a natural choice for the companion knots. Then the above complexity bound of $O(n \log n)$ ops generally increases to $O(n \log^2 n)$ based on a numerically unstable algorithm in [P01, Sect. 3.1] and to $2n^2 - n$ based on a stable version of the Horner's algorithm [BF00]. Even the latter cost bound is still dominated at the subsequent stages of the root approximation.

When the root approximations and the companion knots or the input polynomial are updated, one can recompute the matrix C by applying the algorithms above, but let us next examine some alternative updating means.

5.2. Reversion of a Polynomial, Shift of the Variable, and Their Affect on the GC Matrices

We reverse the input polynomial $c(x)$ in (2.2) and shift the variable x by a scalar s when we preprocessing the input polynomial and apply some popular root-finders, e.g., Jenkins-Traub's. To update the associated GC matrices for the shifted polynomial $c_s(x) = c(x - s)$, we can re-use the same values $d_1, \ldots, d_n$ at the knots $x = s_j + s$ because $c_s(s_j + s) = c(s_j)$ and $q_s'(s + s_j) = q'(s_j)$ for $q_s(x) = q(x-s)$ and $j = 1, \ldots, n$. For the reverse polynomial $c_{rev}(x) = x^n c(1/x)$ we have $c_{rev}(\frac{1}{s_i}) = s_i^{-n} c(s_i)$, $q'_{rev}(\frac{1}{s_i}) = \prod_{j \neq i}(\frac{1}{s_i} - \frac{1}{s_j}) = (-1)^{n-1} s_i^{2-n} q'(s_i) / \prod_{j=1}^{n} s_j$, $i = 1, \ldots, n$, and so we can update $d_1, \ldots, d_n$ by computing $s_1^{2-n}, \ldots, s_n^{2-n}$ and in addition performing $O(n)$ ops.

Alternatively, we can replace the GC matrix C with C^{-1} or $C - sI$, respectively. We can compute the first column of the matrix $(F_c - sI)^{-1}$ in $O(n)$ ops, due to Theorem 4.1, and we can represent the matrix with this column [C96].

Due to the Sherman-Morrison-Woodbury formula and Theorem 4.1, we obtain the DPR1 representation of the matrix $(C - sI)^{-1}$ by using $O(n)$ ops for any matrix C in (3.4), (3.6), and (3.10). In particular it takes $2n$ divisions, $2n$ multiplications and n additions/subtractions for a DPR1 matrix C in (3.4).

5.3. Deflation of Polynomials and GC Matrices

Suppose we have approximated a root z of a polynomial $c(x)$ in (2.2). Then we can deflate the associated matrices C in (3.1), (3.4), (3.6) and (3.10) preserving their structure.

For the Frobenius matrix in (3.1), we just compute the quotient polynomial $c^{new}(x) = \frac{c(x)}{x-z}$ by using $n-1$ subtractions and $n-1$ divisions. For the three other matrix classes we also use $O(n)$ ops but involve no coefficients of $c(x)$ unlike the Frobenius case.

For the DPR1 matrix in (3.4), we replace the vector $\mathbf{s} = (s_i)_{i=1}^{n}$ with $\mathbf{s}^{new} = (s_i)_{i=1}^{n-1}$ and compute the associated vector $\mathbf{d}^{new} = (d_i^{new})_{i=1}^{n-1}$ according to equations (6.2) in [BGP02/04], that is,

$$d_i^{new} = d_i \frac{s_i - s_n}{s_i - z}, \ i = 1, \dots, n-1. \tag{5.1}$$

This takes $2n-2$ additions/subtractions, $n-1$ multiplications, and $n-1$ divisions. If $z \approx s_n$, then $d_i^{new} \approx d_i$ for $i < n$, and we yield cost-free deflation.

Similarly we deflate the matrices C in (3.6) and (3.10). Under (3.10) we write $\overline{\mathbf{s}}^{new} = (\overline{s}_i)_{i=1}^{n-1}$, rely on (3.8), and compute the associated vector $\overline{\mathbf{d}}^{new} = (\overline{d}_i^{new})_{i=1}^{n-1}$ according to the following equations, which extend equations (5.1),

$$\overline{d}_1^{new} = \overline{d}_1 \frac{\overline{s}_n}{z}, \ \overline{d}_i^{new} = \overline{d}_i \frac{\overline{s}_i - \overline{s}_n}{\overline{s}_i - z}, \ i = 2, \dots, n-1. \tag{5.2}$$

The computations involve $2n-3$ additions/subtractions, $n-1$ multiplications, and $n-1$ divisions. We can keep the deflation processes (5.1), (5.2) in the field of real numbers for polynomials with real coefficients. We just need to deflate the pair of the complex conjugate roots as soon as one of them is approximated.

And again if $z \approx \overline{s}_n$, then $\overline{d}_i^{new} \approx \overline{d}_i$ for all $i < n$, and we yield cost-free deflation.

5.4. Updating the Companion Knots and Matrices

If we have updated a single companion knot s_i, we can update the DPR1 matrix in (3.4) by using $O(n)$ ops. Indeed the values $c(s_j)$ remain invariant for $j \neq i$, whereas we can compute the values $c(s_i)$ and $q_i(s_i)$ by using $4n-3$ ops with Horner's algorithm, and we can compute $q_j^{new}(s_j) = q_j^{old}(s_j) \frac{s_j - s_i^{new}}{s_j - s_i^{old}}$ for every $j \neq i$ by using four ops per value.

Similar observations apply to the RBDPR1 and the arrow-head matrices.

6. Root-Finding via Eigen-Solving

6.1. Approximating the Extremal Eigenvalues

In Table 3 we display the numbers of basic operations required at the kth iteration step in four popular eigen-solvers. They approximate the extremal eigenvalues, that is, the eigenvalues which are the farthest from and the closest to the selected shift value s and which for $s = 0$ are the absolutely largest and the absolutely smallest eigenvalues, respectively. Tables 1–3 together furnish us with the respective ops estimates for these eigen-solvers.

TABLE 3. The numbers of multiplications of the matrix C, C^H and $(C-\mu I)^{-1}$ by vectors and additional ops at the kth iteration step

Eigen-solver	$C*v$	C^H*v	$(C-\mu I)^{-1}*v$	Additional Ops
Arnoldi	1			$(4k+4)+O(1)$
non-Hermitian Lanczos	1	1		$15n+O(1)$
Jacobi-Davidson	1		1	$(9+k^2)n+O(1)$
IPI	1		1	$5n-1$

The inverse power iteration (IPI) approximates the single eigenvalue closest to the shift value s. We refer the reader to [GL96, Sects. 8.2.2 and 8.2.3], [S98, Sect. 2.1.2]), and [BDDRvV00], and the bibliography therein on this iteration and its Rayleigh-Ritz block version for approximating some blocks of the extremal eigenvalues. A new modification of the IPI is proposed in [PIMa], whereas the papers [BGP02/04] and [P05] specialize the IPI to the DPR1 and Frobenius input matrices. By applying the IPI to such matrices for the reverse polynomial $c_{rev}(x)$, we approximate the absolutely largest roots of the polynomial $c(x)$.

The Jacobi-Davidson algorithms also approximate the single extremal eigenvalue or a block of such eigenvalues [S98, Sect. 6.2], [BDDRvV00], whereas the Arnoldi and the non-Hermitian Lanczos algorithms [GL96, Sect. 9.4], [S98, Chap. 5], [BDDRvV00] approximate simultaneously a small number of eigenvalues consisting of both eigenvalues closest to and farthest from a fixed shift value. Actually all these algorithms approximate the Ritz eigenpairs, that is, the pairs of the eigenvalues and the associated eigenvectors (or more generally, blocks of the eigenvectors and the associated eigenspaces).

Table 3 does not cover the ops required for approximating a Ritz pair for an $k \times k$ auxiliary Hessenberg (resp. tridiagonal) matrix in the Arnoldi (resp. non-Hermitian Lanczos) algorithm and for computing the Euclidean vector norms (at most two norms are required per step). Actually, to make the Arnoldi and the Jacobi-Davidson algorithms competitive, one must keep k smaller, although such a policy is in conflict with the task of approximating the eigenvalues closely. This seems to give upper hand to the Lanczos algorithm and the IPI.

Another crucial factor is the number of iteration steps required for convergence, but all the cited eigen-solvers have good local and global convergence according to the extensive empirical evidence and partly to the theory [GL96, S98, BDDRvV00]. Local convergence of the Arnoldi and Lanczos algorithms can be substantially speeded up with the shift-and-invert techniques [S98, pages 334–336].

Convergence of the IPI can be additionally accelerated in the case of the Frobenius input matrix $C = F_c$ [P05]. Formally, let $\theta = \max_{\mu\neq\lambda} |\frac{\mu-s}{\lambda-s}|$ where s is the selected shift value approximating an eigenvalue λ, and the maximum is over all other eigenvalues μ. Then the eigenvalue λ is approximated within the error in $O(\theta^k)$ in k IPI steps, whereas the much smaller error bound in $O(\theta^{2^k})$ can be reached in k steps of the algorithm in [P05]. The latter algorithm uses almost as many ops per step as six FFT's at 2^h points for $h = \lceil \log_2(2n-1) \rceil$, that is, the order of $n \log n$ ops per step, versus $O(n)$ ops per an IPI step.

Finally, since all of the above algorithms approximate the eigenvalues which are the closest to the shift value s, a by-product of their application is a *proximity test* at the complex point s for the roots of the polynomial $c(x)$. We exploit this observation at the very end of the section.

6.2. Approximating All Eigenvalues

To extend the algorithms in the previous subsection to computing all eigenvalues, we can recursively combine them with deflating the polynomial $c(x)$ and/or updating its GC matrix (see Sect. 5.3) as long as we can approximate the eigenvalues closely enough to counter the error propagation. We discuss how to improve the initial approximations to the eigenvalues in the next subsections.

We can dispense with deflation and apply the selected eigen-solvers to the same matrix but vary the shift values s trying to direct the eigen-solver to a new eigenvalue. The iteration can occasionally converge to the same eigenvalue already approximated, but according to the empirical evidence and some theory available for Newton's iteration, running it for the order of n to $n \log n$ initial shift values equally spaced on a large circle is usually sufficient to approximate all eigenvalues.

Furthermore, the algorithm in [P05] always enforces convergence to a new eigenvalue of the Frobenius matrix F_c, so that in n applications it outputs approximations to all n eigenvalues.

Finally we recall that the QR algorithm approximates all eigenvalues of a matrix in roughly $10n^3$ ops according to extensive empirical evidence [GL96, Sect. 7.5.6]. The bound relies on using $10n^2$ ops per QR iteration step for an $n \times n$ Hessenberg input matrix. For a DPR1 input and the initial companion knots equally spaced on a circle, as well as for any set of companion knots on a circle or a line, the QR algorithms in [BGP03/05, BGP04] use at most $120n$ ops per step, so that we can extrapolate the cited empirical cost bound to at most $120n^2$ for all eigenvalues.

6.3. Eigen-Solvers and Root-Finders as Root-Refiners

Based on our study in the previous sections, we should approximate the roots of a polynomial $c(x)$ in (2.2) by applying selected eigen-solvers to appropriate GC matrices, performing the computations numerically, with double precision, updating the matrices when the approximations to the eigenvalues improve, and possibly changing the eigen-solvers during the iteration process. As we mentioned in the introduction, one can expect that a variant of this approach with the QR algorithm and the DPR1 GC matrices in [MV95, MV95a, F01/02] should rapidly improve approximations to the eigenvalues to the desired level.

Based on our study in Sect. 3, we should expect the same effect if we use the RBDPR1 (in the real case) or arrow-head matrices instead of the DPR1 matrices. Furthermore, all other eigen-solvers in the previous subsections can be applied instead of the QR algorithm, and next we briefly compare them with each other and with popular polynomial root-finders applied as root-refiners. We must, however, exclude the algorithm in [P05], which is applied to the Frobenius matrix F_c and is not updated when we update the computed approximations to the roots.

The QR algorithm in [BGP03/05] and [BGP04] requires quadratic time per step and quadratic memory space for DPR1 matrices with general complex companion knots and thus becomes inferior as an eigen-refiner.

The IPI and the non-Hermitian Lanczos algorithms seem to be better candidates to be the GC eigen-refiner of choice because they require fewer ops per an iteration step than the Jacobi-Davidson and the Arnoldi algorithms (see Sect. 6.1). There is a potential competition from the popular root-finders applied as root-refiners. They have superlinear local convergence, like the IPI, but require extended precision of computing. Note another practical advantage of the IPI over the popular polynomial root-finders. For a real input matrix C the IPI can be easily extended to confine the computations to the real field. Namely, we should just apply the power iteration step to the real matrix $(sI-C)^{-1}(s^*I-C)^{-1}$ where s and s^* denote two complex conjugate approximations to two complex conjugate eigenvalues of the matrix C.

Tables 4 and 5 display some relevant data on some most popular root-finders that approximate one root at a time and simultaneously all roots, respectively. Note the respective increase of the arithmetic cost per step in Table 5.

Table 4. Four root-finders for a polynomial of a degree n approximating one root at a time (In Müller's and Laguerre's algorithms computing a square root is counted as an op.)

Root-finder	References	ops/step	Order of convergence
Müller's	[T64, pages 210–213], [W68]	$2n+20$	1.84
Newton's	[M73, MR75, NAG88]	$4n$	2
Halley's	[OR00, ST95]	$6n$	3
Laguerre's	[HPR77, P64]	$6n+6$	3

Table 4 does not cover the Jenkins-Traub algorithm in [JT70, JT72]. The statistics of its application show that its performance is similar to the other root-finders in Table 4 (in fact they tend to be inferior in accuracy to the QR based root-finders), but the formal data on its ops count are hard to specify because this algorithm combines various other methods.

Among modifications of the listed root-finders, we note application of Müller's algorithm to the ratio $\frac{c(x)}{c'(x)}$ rather than to the polynimial $c(x)$. This increases the

Table 5. Two root-finders approximating simultaneously all roots of a polynomial of a degree n

Root-finder	References	ops/step	Order of convergence
Durand-Kerner's	[W03, D60, K66]	$(4n-1)n$	2
Aberth's	[B-S63, E67, A73, BF00]	$(7n-3)n$	3

ops count per step to $4n + O(1)$ but substantially improves convergence according to our extensive tests.

6.4. Flowcharts for Root-Finding with Eigen-Solving

To summarize, here is a flowchart of our root-finding for a polynomial $c(x)$ in (2.2).

- *Initial approximation.*

  Select and compute a GC matrix for $c(x)$ (cf. Sect. 5.1).

  Select and apply an eigen-solver for this matrix to compute n distinct approximations to the roots of $c(x)$ (see Sects. 6.1 and 6.2).
- *Updating the GC matrix and the approximations to the roots.*

  Choose the companion knots equal to the computed approximations to the roots and update the GC matrix (cf. Sect. 5.4).

  Apply the IPI n times with the shifts into the n current companion knots to improve the approximations to all eigenvalues.

  Repeat recursively until convergence.

In a modified version of this flowchart, we select a root-finder in Tables 4 or 5 and substitute it for the IPI at the initial and/or updating stage.

Computations in both original and modified versions can include deflation (see Sect. 5.3).

Implementing the flowchart, we should numerically stabilize both eigensolvers (by means of diagonal scaling (see the end of Sect. 3.5)) and root-finders (by means of shifting the variable x to turn the coefficient c_{n-1} into zero). Then we should scale both the variable x and the polynomial $c(x)$, that is, shift to the polynomial $d^n c(x/d)$ for a scalar d chosen to decrease the disparity in the magnitudes of the coefficients of the latter polynomial.

6.5. Divide-and-Conquer Root-Refining and Bounding the Output Errors

We can accelerate root-finding and eigen-solving if we can split a polynomial $c(x)$ into the product $\prod_{i=1}^{k} c_i(x)$ of $k > 1$ nonscalar polynomials $c_i(x)$ and repeat this step recursively (see [P01/02, BP98] and the bibliography therein). Effective splitting algorithms in [S82, K98, P01/02, BGM02] compute the factors $c_i(x)$ in nearly optimal arithmetic and Boolean time provided we know some sufficiently wide root-free annuli on the complex plane that isolate the root sets of the factors $c_i(x)$ from each other (see also [C96, BP96] on some alternative splitting algorithms and [W69, BJ76, B83, DM89, DM90, VD94] on various applications to signal and image processing). The algorithms in [P01/02] compute the desired annuli also in nearly optimal time but are quite involved, which diminishes their practical value. For a large input class, however, the annuli are readily available as by-product of approximating the roots even with a low precision.

With the GC representations in Sects. 3.2 and 3.4 we can bound the approximation errors and detect the basic root-free annuli for splitting based on the following result for the DPR1 and arrow-head matrices.

Theorem 6.1. *The union $\sum_{i=1}^{n} D_i$ (resp. $\sum_{i=1}^{n} \overline{D}_i$) contains all eigenvalues of the matrix C in equation* (3.4) *(resp. the matrix $\overline{C}$ in* (3.10)*) provided D_i (resp. $\overline{D}_i$) denote the discs $\{x : \ |x - s_i + d_i| \le \sum_{j \ne i} |u_j v_i|\}$ or $\{x : \ |x - s_i + d_i| \le \sum_{j \ne i} |u_i v_j|\}$, $i = 1, \dots, n$ (resp. the discs $\{x : \ |x - \overline{s}_1 + \overline{u}_1| \le \sum_{i=2}^{n} |\overline{u}_i|\}$, $\{x : \ |x - \overline{s}_j| \le |\overline{v}_j|\}$, $j = 2, \dots, n$, or the discs $\{x : \ |x - \overline{s}_1 + \overline{u}_1| \le \sum_{j=2}^{n} |\overline{v}_j|\}$, $\{x : \ |x - \overline{s}_i| \le |\overline{u}_i|\}$, $i = 2, \dots, n$). Moreover, if the union of any set of k discs D_i (resp. $\overline{D}_i$) is isolated from all remaining $n - k$ discs, then this union contains exactly k eigenvalues of the matrix C (resp. $\overline{C}$).*

Proof. The theorem (due to [E73] for DPR1 matrices) immediately follows from the Gerschgörin theorem [GL96, Theorem 7.2.1] applied to the matrices C and $\overline{C}$. $\square$

We need $3n - 1$ ops to compute the radii of the discs $D_1, \dots, D_n$ (or just $2n-1$ ops under the choice of parameters in Example 3.1), and we only need $n-1$ ops to compute the radii of the discs $\overline{D}_1, \dots, \overline{D}_n$.

Similarity transforms into a DPR1 matrix (see Sect. 3.6) enable us to extend the estimates in Theorem 6.1 to the RBDPR1 matrices C in (3.6), and we can yield a similar extension from the arrow-head matrices.

All discs D_i (resp. $\overline{D}_i$) are isolated from each other for all i if the matrix C (resp. $\overline{C}$) has n distinct eigenvalues and if the values $|u_i|$ and $|v_i|$ (resp. $|\overline{u}_i|$ and $|\overline{v}_i|$) are small enough. In this case the disc radii serve as upper bounds on the errors of the computed approximations s_i (resp. $\overline{s}_i$) to the eigenvalues.

Finally recall that a proximity test at a point s for the roots of a polynomial $c(x) = \prod_{j=1}^{n}(x - z_j)$ defines a root-free disc $\{x : |x - s| < \min_j |z_j - z|\}$ and that such a proximity test is a by-product of the application of either of the IPI, Arnoldi, non-Hermitian Lanczos and Jacobi-Davidson algorithms to the matrix $sI - C_c$. Now if the latter disc covers the intersection of two discs D_h and D_i (resp. $\overline{D}_h$ and $\overline{D}_i$), then they are isolated from one another. This observation combined with Theorem 6.1 suggests a promising heuristic method for isolating the eigenvalues.

7. Extension to Eigen-Solving

We can extend our eigen-solvers for GC matrices to the matrices A for which we can readily compute the following scalars and vectors.

- the scalar $c_A(x) = \det(xI - A)$ for a scalar x
- the scalars $c'_A(x) = -\operatorname{trace}(xI - A)^{-1} c_A(x)$ and $c''_A(x)$ for a scalar x
- the vector $(xI - A)^{-1}\mathbf{v}$ for a vector $\mathbf{v}$
- the vector $A\mathbf{v}$ for a vector $\mathbf{v}$.

Furthermore, as soon as we have n values $c_A(x)$ at n distinct points $s_1, \dots, s_n$ computed, we can compute GC matrices $C = C_c$ in Sects. 3.2–3.4 for $c(x) = c_A(x)$. Then we can apply our algorithms to compute approximations $\tilde{z}_1, \dots, \tilde{z}_n$ to the roots, which are generally crude due to the rounding errors in computing the GC matrix. We can, however, refine the approximations by applying the IPI or the algorithm in [P05] to the matrix A and the shift values $\tilde{z}_1, \dots, \tilde{z}_n$.

Moreover, we can compute some crude initial approximations to the roots without computing a GC matrix. Indeed, apply the eigen-solvers in Table 3 as long as you compute the vectors $\mathbf{v}$ and apply the root-finders in Tables 4 and 5 as long as you compute the scalars x. In fact the Durand-Kerner's and Müller's algorithms only require the computation of the scalars $c_A(x)$.

For many important classes of matrices all or most of the listed scalars and vectors can be readily computed at a low cost. This is the case, e.g., for various structured (e.g., Toeplitz) matrices [P01, Chap. 5], for banded matrices B having a small bandwidth (e.g., tridiagonal matrices) or more generally, for matrices associated with graphs that have small separator famillies [LRT79, GH90, GS92, PR93].

8. Polynomial and Secular Equations

The polynomial equations $c(x) = 0$ are closely related to the secular equations, encountered in updating the singular value decomposition of a matrix, the solution of the least-squares constrained eigenproblem, invariant subspace computation, divide-and-conquer algorithms for the tridiagonal Hermitian eigenproblem, and the "escalator method" for matrix eigenvalues (see [G73, M97] and the bibliography therein).

For a matrix C in (3.10), recall the characteristic equation $c_C(x) = 0$, rewrite it as $c_C(x) = (x + a)\overline{q}(x) - \sum_{i=2}^{n} \overline{d}_i \overline{q}_i(x) = 0$, for the scalar $a = \overline{u}_1 - \overline{s}_1$ and then divide it by $\overline{q}(x)$ to arrive at the secular equation

$$x + a - \sum_{i=2}^{n} \frac{\overline{d}_i}{x - \overline{s}_i} = 0,$$

whose roots are given by the eigenvalues of the matrix C in (3.10). Likewise, recall the Lagrange interpolation formula $c_C(x) = q(x) + \sum_{i=1}^{n} d_i q_i(x)$ and divide its both sides by $q(x)$ to arrive at the secular equation

$$1 + \sum_{i=1}^{n} \frac{d_i}{x - s_i} = 0,$$

Table 6. Five root-finders for a function $r(\lambda)$

Method	References	Max order i of $r^{(i)}(\lambda)$	Convergence order
Müller's	[T64, pages 210–213], [W68]	0	1.84
Newton's modified	[M73, MR75, NAG88]	1	2
Halley's	[OR00, ST95]	2	3
Laguerre's	[HPR77, P64]	2	3
Laguerre's discrete	[DJLZ96, DJLZ97, Z99]	0	3

whose roots are equal to the eigenvalues of the matrix C in (3.4). By allowing to scale the equation, we reduce the root-finding for any secular equation of the form

$$\alpha x + \beta + \sum_{i=1}^{k} \frac{d_i}{x - s_i} = 0$$

to solving the eigenproblem for the arrow-head or DPR1 matrices. This also enables simple reduction of the polynomial and secular equations to one another.

Appendix. Simplification of Root-Finding

The efficiency of the known polynomial root-finders applied as root-refiners typically decreases where the roots are multiple. Since $\frac{c'(x)}{c(x)} = \sum_{i=1}^{k} \frac{m_i}{x-z_i}$ for $c(x) = \prod_{i=1}^{k}(x - z_i)^{m_i}$ where $z_1, \ldots, z_k$ are distinct, this suggests the application of root-finders to the rational function $\frac{c(x)}{c'(x)}$ or to the polynomial $\frac{c(x)}{g(x)}$ where $g(x) = \gcd(c', c)$ is the gcd of $c'(x)$ and $c(x)$. With approximate division by approximate gcds, we can also replace root clusters by their single simple representatives.

The problem of computing approximate gcds of univariate polynomials is of high independent interest (see [CGTW95, P98/01, GKMYZ04, Za, LYZ05] and the bibliography therein). The approach in [P98/01] remains a good candidate for being the method of choice. It relies on the reduction to polynomial root-finding. Can any further progress be obtained based on matrix methods, e.g., on Theorem 6.1?

To avoid vicious circle of the back-and-forth transition between root-finding and approximate gcds, we can apply the root-finders to the rational function $f(x) = \frac{c(x)}{c'(x)}$. The Börsch-Supan's root-finder [B-S63] (widely known as Aberth's or Ehrlich's [B96]) proceeds by recursively computing the values of this function. The iterative processes in Tables 4 and 5 can be reduced essentially to the recursive evaluation of $c^{(i)}(x)$ at the approximation points x for $i = 0, 1, \ldots, k$ and a small fixed integer k. The poles of the function $\frac{c(x)}{c'(x)}$ can cause divergence, but overall convergence tends to be faster and more reliable when we apply Müller's method to the function $f(x)$ according to our tests with Müller's and Newton's root-finders, each applied to both $c(x)$ and $f(x)$.

References

[A73] O. Aberth, Iteration Methods For Finding All Zeros of a Polynomial Simultaneously, *Math. Comp.*, **27**, **122**, 339–344, 1973.

[B75] S. Barnett, A Companion Matrix Analogue for Orthogonal Polynomials, *Linear Algebra and Its Applications*, **12**, **3**, 97–208, 1975.

[B83] S. Barnett, *Polynomials and Linear Control Systems*, Marcel Dekker, New York, 1983.

[B96] D. A. Bini, Numerical Computation of Polynomial Zeros by Means of Aberth's Method, *Numerical Algorithms*, **13**, **3–4**, 179–200, 1996.

[BBCD93] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[BDDRvV00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[BF00] D. A. Bini, G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**, 127–173, 2000.

[BGM02] D. A. Bini, L. Gemignani, B. Meini, Computations with Infinite Toeplitz Matrices and Polynomials, *Linear Algebra and Its Applications*, **343–344**, 21–61, 2002.

[BGP02/04] D. A. Bini, L. Gemignani, V. Y. Pan, Inverse Power and Durand/Kerner Iteration for Univariate Polynomial Root-finding, *Computers and Mathematics (with Applications)*, **47**, **2/3**, 447–459, 2004. (Also Technical Report TR 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002.)

[BGP04] D. A. Bini, L. Gemignani, V. Y. Pan, Improved Initialization of the Accelerated and Robust QR-like Polynomial Root-finding, *Electronic Transactions on Numerical Analysis*, **17**, 195–205, 2004. Proc. version in *Proceedings of the Seventh International Workshop on Computer Algebra in Scientific Computing (CASC ?4)*, St. Petersburg, Russia (July 2004), (edited by E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov), 39-50, Technische Univ. München, Germany, 2004.

[BGP03/05] D. A. Bini, L. Gemignani, V. Y. Pan, Fast and Stable QR Eigenvalue Algorithms for Generalized Companion Matrices and Secular Equation, *Numerische Math.*, **3**, 373–408, 2005. (Also Technical Report 1470, *Department of Math., University of Pisa*, Pisa, Italy, July 2003.)

[BJ76] G. E. P. Box, G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, California, 1976.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[BP96] D. Bini, V. Y. Pan, Graeffe's, Chebyshev, and Cardinal's Processes for Splitting a Polynomial into Factors, *J. Complexity*, **12**, 492–511, 1996.

[BP98] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real, *SIAM Journal on Computing*, **27**, **4**, 1099–1115, 1998. Proc. Version: Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem, *in Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA?1)*, 384-393, ACM Press, New York, and SIAM Publications, Philadelphia, January 1991.

[B-S63] W. Börsch-Supan, A-posteriori Error Bounds for the Zeros of Polynomials, *Numerische Math.*, **5**, 380–398, 1963.

[C91] C. Carstensen, Linear Construction of Companion Matrices, *Linear Algebra and Its Applications*, **149**, 191–214, 1991.

[C96] J. P. Cardinal, On Two Iterative Methods for Approximating the Roots of a Polynomial, *Lectures in Applied Mathematics*, **32** (*Proceedings of AMS-SIAM Summer Seminar: Mathematics of Numerical Analysis: Real Number Algorithms* (J. Renegar, M. Shub, and S. Smale, editors), Park City, Utah, 1995), 165–188, American Mathematical Society, Providence, Rhode Island, 1996.

[CGTW95] R. M. Corless, P. M. Gianni, B. M. Trager, S. M. Watt, The Singular Value Decomposition for Polynomail Systems, *Proc. Intern. Symposium on Symbolic and Algebriac Computation* (ISSAC'95), 195–207, ACM Press, New York, 1995.

[CN94] D. Coppersmith, C. A. Neff, Roots of a Polynomial and Its Derivatives. *Proc. of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'94), 271–279, ACM Press, New York, and SIAM Publications, Philadelphia, 1994.

[D60] E. Durand, Solutions numériques des équations algébriques, *Tome 1: Equations du type F(X)=0; Racines d'un polynôme*, Masson, Paris, 1960.

[DJLZ96] Q. Du, M. Jin, T. Y. Li, Z. Zeng, Quasi-Laguerre Iteration in Solving Symmetric Tridiagonal Eigenvalue Problems. *SIAM J. Sci. Comput.*, **17, 6**, 1347–1368, 1996.

[DJLZ97] Q. Du, M. Jin, T. Y. Li, Z. Zeng, The Quasi-Laguerre Iteration. *Math. of Computation.* **66, 217**, 345–361, 1997.

[DM89] C. J. Demeure, C. T. Mullis, The Euclid Algorithm and Fast Computation of Cross-Covariance and Autocovariance Sequences, *IEEE Trans. Acoust., Speech and Signal Processing*, **37**, 545–552, 1989.

[DM90] C. J. Demeure, C. T. Mullis, A Newton-Raphson Method for Moving-Average Spectral Factorization Using the Euclid Algorithm, *IEEE Trans. Acoust., Speech and Signal Processing*, **38**, 1697–1709, 1990.

[E67] L. W. Ehrlich, A Modified Newton Method for Polynomials, *Comm. of ACM*, **10**, 107–108, 1967.

[E73] L. Elsner, A Remark on Simultaneous Inclusions of the Zeros of a Polynomial by Gershgörin's Theorem, *Numerische Math.*, **21**, 425–427, 1973.

[EMP04] E. Z. Emiris, B. Mourrain, V. Y. Pan, Guest Editors, Algebraic and Numerical Algorithms, *Special Issue of Theoretical Computer Science*, **315, 2–3**, 307–672, 2004.

[EP02] I. Z. Emiris, V. Y. Pan, Symbolic and Numerical Methods for Exploiting Structure in Constructing Resultant Matrices, *J. of Symbolic Computation*, **33**, 393-413, 2002.

[F90] M. Fiedler, Expressing a Polynomial As the Characteristic Polynomial of a Symmetric Matrix, *Linear Algebra and Its Applications*, **141**, 265–270, 1990.

[F01/02] S. Fortune, An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials, *J. of Symbolic Computation*, **33, 5**, 627–646, 2002. Proc. version in *Proc. Intern. Symp. on Symbolic and Algebraic Computation (ISSAC'01)*, 121–128, ACM Press, New York, 2001.

[G52/58] A. Gel'fond, *Differenzenrechnung*, Deutsher Verlag Der Wissenschaften, Berlin, 1958. (Russian edition: Moscow, 1952.)

[G73] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Review*, **15**, 318–334, 1973.

[GH90] J. R. Gilbert, H. Hafsteinsson, Parallel Symbolic Factorization of Sparse Linear Systems, *Parallel Computing*, **14**, 151–162, 1990.

[GKMYZ04] S. Gao, E. Kaltofen, J. May, Z. Yang, S. Zhi, Approximate Factorization of Multivariate Polynomial via Differential Equations, *Proc. International Symposium on Symbolic and Algebraic Computaion (ISSAC'04)*, 167–174, ACM Press, New York, 2004.

[GL96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[GS92] J. R. Gilbert, R. Schreiber, Highly Parallel Sparse Cholesky Factorization, *SIAM J. on Scientific Computing*, **13**, 1151–1172, 1992.

[HPR77] E. Hansen, M. Patrick, J. Rusnack, Some Modification of Laguerre's Method, *BIT*, **17**, 409–417, 1977.

[JT70] M. A. Jenkins, J. F. Traub, A Three-Stage Variable-Shift Iteration for Polynomial Zeros and Its Relation to Generalized Rayleigh Iteration, *Numerische Math.*, **14**, 252–263, 1969/1970.

[JT72] M. A. Jenkins, J. F. Traub, A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration, *SIAM J. on Numerical Analysis*, **7**, 545–566, 1970.

[JV04] J. F. Jónsson, S. Vavasis, Solving Polynomials with Small Leading Coefficients, *SIAM J. on Matrix Analysis and Applications*, **26, 2**, 400–412, 2004.

[K66] I. O. Kerner, Ein Gesamtschrittverfahren zur Berechung der Nullstellen von Polynomen, *Numerische Math.*, **8**, 290–294, 1966.

[K98] P. Kirrinnis, Polynomial Factorization and Partial Fraction Decomposition by Simultaneous Newton's Iteration, *J. of Complexity*, **14**, 378–444, 1998.

[LF94] M. Lang, B. C. Frenzel, Polynomial Root-Finding, *IEEE Signal Processing Letters*, **1**, **10**, 141–143, 1994.

[LRT79] R. J. Lipton, D. Rose, R. E. Tarjan, Generalized Nested Dissection, *SIAM J. on Numerical Analysis*, **16**, **2**, 346–358, 1979.

[LYZ05] B. Li, Z. Yang, L. Zhi, Fast Low Rank Approximation of a Sylvester Matrix by Structured Total Least Norm, *Journal JSSAC*, **11**, 165–174, 2005.

[M73] K. Madsen, A Root-Finding Algorithm Based on Newton's Method,*BIT*, **13**, 71–75, 1973.

[M97] A. Melman, A Unifying Convergence Analysis of Second-Order Methods for Secular Equations, *Math. Comp.*, **66**, 333–344, 1997.

[McN93] J. M. McNamee, Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **47**, 391–394, 1993.

[McN97] J. M. McNamee, A Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **78**, 1, 1997.

[McN99] J. M. McNamee, An Updated Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **110**, 305–306, 1999.

[McN02] J. M. McNamee, A 2002 Updated Supplementary Bibliography on Roots of Polynomials, *J. Computational and Applied Mathematics*, **142**, 433–434, 2002.

[MP00] B. Mourrain, V. Y. Pan, Multivariate Polynomials, Duality and Structured Matrices, *J. of Complexity*, **16**, **1**, 110–180, 2000.

[MR75] K. Madsen, J. Reid, Fortran Subroutines for Finding Polynomial Zeros, Report HL75/1172 (C.13), *Computer Science and Systems Division*, A. E. R. E. Harwell, Oxford, 1975.

[MV95] F. Malek, R. Vaillancourt, Polynomial Zerofinding Iterative Matrix Algorithms, *Computers and Math. with Applications*, **29**, **1**, 1–13, 1995.

[MV95a] F. Malek, R. Vaillancourt, A Composite Polynomial Zerofinding Matrix Algorithm, *Computers and Math. with Applications*, **30**, **2**, 37–47, 1995.

[NAG88] *NAG Fortran Library Manual*, Mark 13, Vol. **1**, 1988.

[NR94] C. A. Neff, J. H. Reif, An $O(n^{l+\epsilon})$ Algorithm for the Complex Root Problem, *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Scinece (FOCS'94)*, 540–547, IEEE Computer Society Press, Los Alamitos, California, 1994.

[OR00] J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, 2000.

[P64] B. Parlett, Laguerre's Method Applied to the Matrix Eigenvalue Problem, *Math. of Computation*, **18**, 464–485, 1964.

[P92] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review*, **34**, **2**, 225–262, 1992.

[P95] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symp. on Theory of Computing (STOC'95)*, 741–750, ACM Press, New York, May 1995.

[P96] V. Y. Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros, *Computers and Math. (with Applications)*, **31**, **12**, 97–138, 1996.

[P97] V. Y. Pan, Solving a Polynomial Equation: Some History and Recent Progress, *SIAM Review*, **39, 2**, 187–220, 1997.

[P98] V. Y. Pan, Some Recent Algebraic/Numerical Algorithms, *Electronic Proceedings of IMACS/ACA'98*, 1998.
http:www-troja.fjfi.cvut.cz/aca98/sessions/approximate/pan/

[P98/01] V. Y. Pan, Numerical Computation of a Polynomial GCD and Extensions, *Information and Computation*, **167**, **2**, 71–85, 2001. Proc. version in *Proc. of 9th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'98)*, 68–77, ACM Press, New York, and SIAM Publications, Philadelphia, 1998.

[P00] V. Y. Pan, Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration, *J. of Complexity*, **16**, **1**, 213–264, 2000.

[P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[P01/02] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations*, **33, 5**, 701–733, 2002. Proc. version in *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC ?1),* 253–267, ACM Press, New York, 2001.

[P05] V. Y. Pan, Amended DSeSC Power Method for Polynomial Root-finding, *Computers and Math. with Applications*, **49, 9–10**, 1515–1524, 2005.

[PIMa] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, I. Taj-Eddin, Y. Tang, X. Yan, Additive Preconditioning and Aggregation in Matrix Computations, *Computers and Math. with Applications*, in press.

[PMRTa] V. Y. Pan, B. Murphy, R. E. Rosholt, Y. Tang, Real Root-Finding, submitted to *Computers and Math. (with Applications)*.

[PMRTYCa] V. Y. Pan, B. Murphy, R. E. Rosholt, Y. Tang, X. Yan, W. Cao, Linking Arrow-head, DPR1, and TPR1 Matrix Structures, preprint, 2005. Proc. version (by V. Y. Pan) in Proc. of *Annual Symposium on Discrete Algorithms (SODA'05)*, 1069–1078, ACM Press, New York, and SIAM Publications, Philadelphia, 2005.

[PR93] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing*, **22**, **6**, 1227–1250, 1993.

[S82] A. Schönhage, The Fundamental Theorem of Algebra in Terms of Computational Complexity, *Mathematics Department, University of Tübingen*, Germany, 1982.

[S98] G. W. Stewart, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, 1998.

[ST95] T. R. Scavo, J. B. Thoo, On the Geometry of Halley's Method., *Amer. Math. Monthly*, **102**, 417–426, 1995.

[T64] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

[VD94] P. M. Van Dooren, Some Numerical Challenges in Control Theory. In *Linear Algebra for Control Theory*, Volume **62** of IMA Vol. Math. Appl., Springer, 1994.

[W03] K. Weierstrass, Neuer Beweis des Fundamentalsatzes der Algebra, *Mathematische Werker*, Tome **III**, Mayer und Müller, Berlin, 251–269, 1903.

[W65] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[W68] V. Whitley, Certification of Algorithm 196: Müller's Method for Finding Roots of Arbitrary Function, *Comm. ACM*, **11**, 12–14, 1968.

[W69] G. T. Wilson, Factorization of the Covariance Generating Function of a Pure Moving-average Process, *SIAM J. Num. Anal.*, **6**, 1–7, 1969.

[Z99] X. Zou, Analysis of the Quasi-Laguerre Method, *Numerische Math.*, **82**, 491–519, 1999.

[Za] Z. Zeng, The Approximate GCD of Inexact Polynomials, Part I: a Univariate Algorithm, preprint, 2004.

Victor Y. Pan, Brian Murphy and Rhys Eric Rosholt
Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468, USA
http://comet.lehman.cuny.edu/vpan/
e-mail: `victor.pan@lehman.cuny.edu`
`brian.murphy@lehman.cuny.edu`
`rhys.rosholt@lehman.cuny.edu`

Dmitriy Ivolgin, Yuqing Tang and Xiaodong Yan
Ph.D. Program in Computer Science
Graduate Center of the City University of New York
New York, NY 10036 USA
e-mail: `divolgin@gc.cuny.edu`
`ytang@gc.cuny.edu`
`xyan@gc.cuny.edu`

Xinmao Wang
Ph.D. Program in Mathematics
Graduate Center of the City University of New York
New York, NY 10036 USA
Present address:
Department of Mathematics,
University of Science and Technology of China,
Hefei, Anhui 230026, China
e-mail: `xinmao@ustc.edu.cn`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 211–222

Galois Theory via Eigenvalue Methods

David A. Cox

Abstract. In recent years, eigenvalue and eigenvector methods have played an increasingly important role in solving polynomial systems. This expository article explains how these ideas can be applied to Galois theory.

Introduction

Galois theory was invented to understand the solutions of a univariate polynomial equation and led significant developments in pure mathematics. More recently, solutions of multivariate polynomial systems have been studied by the eigenvalue and eigenvector methods. This expository article will explore some of the unexpected connections between these approaches. Full details can be found in [4].

1. Eigenvalue Methods

We begin with a quick review of eigenvalue methods. Given a system of polynomial equations

$$f_1(x_1,\ldots,x_n) = \cdots = f_s(x_1,\ldots,x_n) = 0 \tag{1.1}$$

with coefficients in an infinite field F, we get the quotient ring

$$A = F[x_1,\ldots,x_n]/\langle f_1,\ldots,f_s\rangle,$$

which is an algebra over F since it is both a ring and a vector space over F in a compatible way. Every polynomial $h \in F[x_1,\ldots,x_n]$ gives a *multiplication map*

$$m_h : A \longrightarrow A$$

defined by $m_h([f]) = [hf]$, where $[f] \in A$ is the coset of f in the quotient ring A. It is well-known that A is finite-dimensional over F if and only if the equations (1.1) have only finitely many solutions over the algebraic closure $\overline{F}$.

The following *Eigenvalue Theorem* was noticed by Lazard [11] in 1981 and developed in the context of resultants by Auzinger and Stetter [2] in 1988.

Theorem 1.1. *Assume that* (1.1) *has a finite positive number of solutions. Then the eigenvalues of m_h are the values of h at the solutions of* (1.1) *over $\overline{F}$.*

Eigenvalue and eigenvector methods have been studied in numerous papers, many of which are listed in references to [4] and [7]. See also Stetter's book [14]. From a symbolic point of view, the numerical information contained in the eigenvalues is encoded into the characteristic polynomial $\mathrm{CharPoly}_{m_h}(u)$ of m_h. This paper will illustrate some surprising uses of these characteristic polynomials.

2. Single-Variable Representation

We will be interested in the case when the multiplication map m_h is *non-derogatory*, meaning that all of its eigenspaces have dimension one. As explained in [4], the eigenvectors of m_h give especially useful information about the solutions of (1.1) when m_h is non-derogatory.

For a solution p, we get the local ring A_p such that $\dim_F A_p$ is the multiplicity of the solution. We now define a special kind of solution using the ring A_p.

Definition 2.1. A solution p of (1.1) is **curvilinear** if $A_p \simeq F[x]/\langle x^k\rangle$ for some integer $k \geq 1$.

Alternatively, let $\mathfrak{m}_p$ be the maximal ideal of A_p. The integer

$$e_p = \dim_F \mathfrak{m}_p/\mathfrak{m}_p^2 = \#\,\text{minimal generators of } \mathfrak{m}_p \tag{2.1}$$

is called the *embedding dimension* of A_p. Then one can prove that a solution p is curvilinear if and only if A_p has embedding dimension $e_p \leq 1$.

The following result is proved in [4].

Theorem 2.2. *There exists $h \in F[x_1, \ldots, x_n]$ such that m_h is non-derogatory if and only if every solution of* (1.1) *is curvilinear. Furthermore, if this happens, then m_h are non-derogatory when h is a generic linear combination of $x_1, \ldots, x_n$.*

One property of the non-derogatory case is that when m_h is non-derogatory, we can represent the algebra A using one variable. Here is the precise result.

Proposition 2.3. *Assume that $h \in F[x_1, \ldots, x_n]$ and that m_h is non-derogatory. Then*

$$F[u]/\langle \mathrm{CharPoly}_{m_h}(u)\rangle \simeq A.$$

Proof. Consider the map $F[u] \to A$ defined by $P(u) \mapsto [P(h)]$. One easily shows that $P(u)$ is in the kernel if and only if $P(m_h)$ is the zero linear tranformation. By the definition of minimial polynomial, it follows that the kernel of this map is generated by $\mathrm{MinPoly}_{m_h}(u)$. Thus we get an injective homomorphism

$$F[u]/\langle \mathrm{MinPoly}_{m_h}(u)\rangle \longrightarrow A.$$

But $\mathrm{MinPoly}_{m_h}(u) = \mathrm{CharPoly}_{m_h}(u)$ since m_h is non-derogatory, and

$$\dim_F F[u]/\langle \mathrm{CharPoly}_{m_h}(u)\rangle = \deg \mathrm{CharPoly}_{m_h}(u) = \dim_F A.$$

It follows that the above injection is the desired isomorphism. □

When all of the solutions are curvilinear (e.g., they all have multiplicity 1), Proposition 2.3 applies when h is a generic linear combination of the variables.

3. Factoring

We will use characteristic polynomials of multiplication maps m_h to do factoring over number fields.

Suppose that $f(x), g(x) \in \mathbb{Q}[x]$ are irreducible with roots $\alpha, \beta \in \mathbb{C}$ such that $f(\alpha) = g(\beta) = 0$. Then $f(x)$ factors into irreducibles over $\mathbb{Q}(\beta)$, say

$$f(x) = f_1(x) \cdots f_r(x), \quad f_i(x) \in \mathbb{Q}(\beta)[x]. \tag{3.1}$$

This is easy to say, but how is this done in practice? We will describe a method due to Kronecker that reduces this problem to factorization over $\mathbb{Q}$ (using known algorithms) and gcd computations in $\mathbb{Q}(\beta)$ (using the Euclidean algorithm).

Consider $A = \mathbb{Q}[x, y]/\langle f(x), g(y)\rangle$. For $h \in \mathbb{Q}[x, y]$, we let $\Phi(u) \in \mathbb{Q}[u]$ denote the characteristic polynomial of $m_h : A \to A$.

Theorem 3.1. *Pick $t \in \mathbb{Q}$ such that $h = x + ty$ takes distinct values at the solutions of $f(x) = g(y) = 0$ and let*

$$\Phi(u) = \prod_{i=1}^{r} \Phi_i(u)$$

be the irreducible factorization of $\Phi(u)$ in $\mathbb{Q}[u]$. Then the irreducible factorization of $f(x)$ over $\mathbb{Q}(\beta)$ is

$$f(x) = f_1(x) \cdots f_r(x),$$

where

$$f_i(x) = \gcd(\Phi_i(x + t\beta), f(x))$$

and the gcd *is computed in $\mathbb{Q}(\beta)[x]$.*

Proof. One easily sees that the solutions of $f(x) = g(y) = 0$ have multiplicity 1 since $f(x)$ and $g(y)$ are separable. By assumption, $h = x + ty$ takes distinct values at all solutions of $f(x) = g(y) = 0$. Since they have multiplicity 1, m_h is non-derogatory. Then the single-variable representation given by Proposition 2.3 implies the map sending u to $[x + ty] \in A$ induces an isomorphism

$$\mathbb{Q}[u]/\langle\Phi(u)\rangle \simeq A$$

since $\Phi(u)$ is the characteristic polynomial of m_h. Since the eigenvalues all have multiplicity 1, the above factorization of $\Phi(u)$ is a product of distinct irreducibles.

By the Chinese Remainder Theorem, this factorization gives a decomposition

$$A \simeq \mathbb{Q}[u]/\langle\Phi(u)\rangle \simeq \prod_{i=1}^{r} \mathbb{Q}[u]/\langle\Phi_i(u)\rangle$$

into a product of fields. Using the definition of A, this transforms into the product of fields given by

$$A = \mathbb{Q}[x,y]/\langle g(y), f(x)\rangle \simeq \prod_{i=1}^{r} \mathbb{Q}[x,y]/\langle g(y), f(x), \Phi_i(x+ty)\rangle. \tag{3.2}$$

Since $y \mapsto \beta$ induces $\mathbb{Q}[y]/\langle g(y)\rangle \simeq \mathbb{Q}(\beta)$, we can rewrite (3.2) as a product of fields

$$A \simeq \prod_{i=1}^{r} \mathbb{Q}(\beta)[x]/\langle f(x), \Phi_i(x+t\beta)\rangle. \tag{3.3}$$

However, $\mathbb{Q}(\beta)[x]$ is a PID, so that $\langle f(x), \Phi_i(x+t\beta)\rangle$ is the principal ideal generated by $f_i(x) = \gcd(f(x), \Phi_i(x+t\beta))$. Since each factor in the product in (3.3) is a field, $f_i(x)$ is irreducible over $\mathbb{Q}(\beta)$. Notice also that $f_i(x)$ divides $f(x)$. Showing that this gives all irreducible factors of $f(x)$ is straightforward—see [4] for details. □

This theorem gives the following algorithm for factoring $f(x)$ over $\mathbb{Q}(\beta)$:

- Pick a random $t \in \mathbb{Q}$ and compute $\Phi(u) = \mathrm{CharPoly}_{m_h}(u)$ for $h = x + ty$. Also compute $\mathrm{Disc}(\Phi(u))$.
- If $\mathrm{Disc}(\Phi(u)) \neq 0$, then factor $\Phi(u) = \prod_{i=1}^{r} \Phi_i(u)$ into irreducibles in $\mathbb{Q}[u]$ and for each i compute $\gcd(\Phi_i(x+t\beta), f(x))$ in $\mathbb{Q}(\beta)[x]$. This gives the desired factorization.
- If $\mathrm{Disc}(\Phi(u)) = 0$, then pick a new $t \in \mathbb{Q}$ and return to the first bullet.

An alternate approach would be to follow what Kronecker does on pages 258–259 of [10, Vol. II] and regard t as a variable in $h = x + ty$. Then $\Phi(u)$ becomes a polynomial $\Phi(u,t) \in \mathbb{Q}[x,t]$. Now factor $\Phi(u,t)$ into irreducibles $\mathbb{Q}[u,t]$, say $\Phi(u,t) = \prod_{i=1}^{r} \Phi_i(u,t)$. Then $f_i(x,\beta)$ can be recovered from $\Phi_i(x+t\beta,t)$. A rigorously constructive version of this is described in [5].

We can also use characteristic polynomials to do primary decomposition. The idea is to look at (3.2) from the point of view of ideals, which easily implies that

$$\langle f(x), g(y)\rangle = \bigcap_{i=1}^{r} \langle f(x), g(y), \Phi_i(x+ty)\rangle.$$

This is the primary decomposition of $\langle f(x), g(y)\rangle$ since the ideals in the intersection on the right are maximal. Hence, in this special case, we can compute a primary decomposition using the characteristic polynomial of a multiplication map. The general case is discussed in [4].

4. Galois Theory

Consider the splitting field of $x^2 - x - 1 \in \mathbb{Q}[x]$. Two simple description of this field are

$$\mathbb{Q}(\sqrt{5}) \quad \text{and} \quad \mathbb{Q}[y]/\langle y^2 - 5\rangle.$$

However, we will see that the splitting field can also be expressed as

$$\mathbb{Q}[x_1, x_2]/\langle x_1 + x_2 - 1, x_1x_2 + 1\rangle. \tag{4.1}$$

Although this may seem more complicated, it has the advantage of giving explicit descriptions of the roots (the cosets of x_1 and x_2) and the Galois action (permute these two cosets). Note also that the generators of ideal appearing in (4.1) make perfect sense since they give the sum and product of the roots of $x^2 - x - 1$.

We will generalize this approach to arbitrary polynomials using the methods developed in earlier sections. Many of the ideas discussed here are well-known to researchers in computational Galois theory. See, for example, [1] and [12].

4.1. Splitting Algebras

Let F be an infinite field and $f(x) \in F[x]$ be a monic polynomial of degree n with distinct roots. We will write $f(x)$ as

$$f(x) = x^n - c_1 x^{n-1} + \cdots + (-1)^n c_n, \quad c_i \in F.$$

Let $\sigma_1, \ldots, \sigma_n$ denote the elementary symmetric polynomials in $x_1, \ldots, x_n$. Then consider the system of n equations in $x_1, \ldots, x_n$ given by

$$\sigma_1(x_1, \ldots, x_n) - c_1 = \cdots = \sigma_n(x_1, \ldots, x_n) - c_n = 0. \tag{4.2}$$

The associated algebra

$$A = F[x_1, \ldots, x_n]/\langle \sigma_1 - c_1, \ldots, \sigma_n - c_n \rangle$$

is the *splitting algebra* of f over F. The system (4.2) and the algebra A were first written down by Kronecker in 1882 and 1887 respectively (see page 282 of [10, Vol. II] for the equations and page 213 of [10, Vol. III] for the algebra). A modern treatment of the splitting algebra appears in the paper [6].

4.1.1. The Universal Property. The natural map $F[x_1, \ldots, x_n] \to A$ takes σ_i to c_i, so that by the identity

$$(x - x_1) \cdots (x - x_n) = x^n - \sigma_1 x^{n-1} + \cdots + (-1)^n \sigma_n,$$

the cosets $[x_i] \in A$ become roots of $f(x)$. Thus $f(x)$ splits completely over A.

But more is true, for the factorization of $f(x)$ over A controls *all possible* splittings of $f(x)$. Here's why. If $f(x)$ splits completely in some F-algebra R, say

$$f(x) = (x - \alpha_1) \cdots (x - \alpha_n), \quad \alpha_1, \ldots, \alpha_n \in R,$$

then $x_i \mapsto \alpha_i$ induces an F-algebra homomorphism $\varphi : A \to R$ such that this splitting is the image under φ of the splitting of $f(x)$ over A. The splitting of $f(x)$ over A is thus "universal" since any splitting is a homomorphic image of this one.

4.1.2. The Dimension of A. We know that $\dim_F A$ is the number of solutions, counted with multiplicity. Let $\overline{F}$ an the algebraic closure of F and fix a splitting

$$f(x) = (x - \alpha_1) \cdots (x - \alpha_n) \in \overline{F}[x].$$

Using this, one easily sees that (4.2) has the $n!$ solutions given by

$$(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}), \quad \sigma \in S_n.$$

We can also determine the multiplicities of these solutions. Since $\sigma_i - c_i$ has degree i as a polynomial in $x_1, \ldots, x_n$, Bézout's theorem tells us that (4.2) has at most

$1 \cdot 2 \cdot 3 \cdots n = n!$ solutions, counting multiplicity. Since we have $n!$ solutions, the multiplicities must all be 1. Hence

$$\dim_F A = n!.$$

4.1.3. The Action of S_n. The symmetric group S_n acts on $F[x_1, \dots, x_n]$ by permuting the variables. Since $\sigma_i - c_i$ is invariant under this action, the action descends to an action of S_n on the splitting algebra A.

4.1.4. The Emergence of Splitting Fields. Although f splits over A, this algebra need not be a field. So how does A relate to the splitting fields of f over F? We will analyze this following Kronecker's approach.

Since F is infinite, $h = t_1x_1 + \cdots + t_nx_n$ takes distinct values at the solutions of (4.2) for most choices of $t_1, \dots, t_n \in F$. Since all solutions of (4.2) have multiplicity 1, the characteristic polynomial of m_h on A is

$$\mathrm{CharPoly}_{m_h}(u) = \prod_{\sigma \in S_n} \bigl(u - (t_1\alpha_{\sigma(1)} + \cdots + t_n\alpha_{\sigma(n)})\bigr) \tag{4.3}$$

and the linear map m_h is non-derogatory. By Proposition 2.3, the map sending u to $[t_1x_1 + \cdots t_nx_n] \in A$ induces an F-algebra isomorphism

$$F[u]/\langle \mathrm{CharPoly}_{m_h}(u)\rangle \simeq A.$$

Now factor $\mathrm{CharPoly}_{m_h}(u)$ into monic irreducible polynomials in $F[u]$, say

$$\mathrm{CharPoly}_{m_h}(u) = \prod_{i=1}^{r} G_i(u).$$

Since $\mathrm{CharPoly}_{m_h}(u)$ has distinct roots, the $G_i(u)$ are distinct. Hence we have isomorphisms

$$A \simeq F[u]/\langle \mathrm{CharPoly}_{m_h}(u)\rangle \simeq \prod_{i=1}^{r} \underbrace{F[u]/\langle G_i(u)\rangle}_{K_i}. \tag{4.4}$$

Each K_i is a field, and since the projection map $A \to K_i$ is surjective, each K_i is a splitting field of f over F. Thus the factorization of the characteristic polynomial of m_h shows that A is isomorphic to a product of fields, each of which is a splitting field of f over F.

4.1.5. History. The methods described here are due to Galois and Kronecker. In 1830 Galois chose $t_1, \dots, t_n$ such that the $n!$ values $t_1\alpha_{\sigma(1)} + \cdots + t_n\alpha_{\sigma(n)}$ are distinct and showed that

$$V = t_1\alpha_1 + \cdots + t_n\alpha_n$$

is a primitive element of the splitting field. He also used the polynomial on the right-hand side of (4.3). Unlike Kronecker, Galois simply assumed the existence of the roots.

In 1887 Kronecker gave the first rigorous construction of splitting fields. He first proved the existence of $t_1, \dots, t_n$ as above and then factored $\mathrm{CharPoly}_{m_h}(u)$

into irreducibles. If $G_i(u)$ is one of the factors of $\mathrm{CharPoly}_{m_h}(u)$, he showed that $F[u]/\langle G_i(u)\rangle$ is a splitting field of f over F.

4.2. Some Galois Theory

We now use the above description of A to prove some standard results in Galois theory. We begin by observing that A has two structures: an action of S_n and a product decomposition

$$A \simeq \prod_{i=1}^{r} K_i,$$

where K_i is a splitting field of f over F. As we will see, the Galois group arises naturally from the interaction between these structures.

Since the decomposition $A \simeq \prod_{i=1}^{r} K_i$ is unique up to isomorphism, it follows that for $1 \le i \le r$ and $\sigma \in S_n$, we have $\sigma(K_i) = K_j$ for some j. This easily implies the first part of the following result.

Proposition 4.1. *For each $i = 1, \ldots, r$, there is a natural isomorphism*

$$\mathrm{Gal}(K_i/F) \simeq \{\sigma \in S_n \mid \sigma(K_i) = K_i\}.$$

Furthermore, S_n acts transitively on the set of fields $\{K_1, \ldots, K_r\}$.

Proof. It remains to prove transitivity. Under the isomorphism

$$F[u]/\langle \mathrm{CharPoly}_{m_h}(u)\rangle \simeq A,$$

S_n permutes the factors of $\mathrm{CharPoly}_{m_h}(u) = \prod_{i=1}^{r} G_i(u)$. Over $\overline{F}$, the factorization becomes

$$\mathrm{CharPoly}_{m_h}(u) = \prod_{\sigma \in S_n} \big(u - (t_1\alpha_{\sigma(1)} + \cdots + t_n\alpha_{\sigma(n)})\big).$$

This shows that S_n must permute the $G_i(u)$ transitively. By (4.4), we conclude that S_n permutes the K_i transitively. □

We can use Proposition 4.1 to prove some classic results of Galois theory as follows. We begin with the uniqueness of splitting fields.

Theorem 4.2. *All splitting fields of f over F are isomorphic via an isomorphism that is the identity on F.*

Proof. Let L be an arbitrary splitting field of f over F. Then splitting of f over L must come from the universal splitting via an F-algebra homomorphism $\varphi : A \to L$. Furthermore, φ is onto since the roots of f generate L over F. Using the decomposition $A \simeq \prod_{i=1}^{r} K_i$, we obtain a surjection

$$\prod_{i=1}^{r} K_i \longrightarrow L.$$

It is now easy to see that $L \simeq K_i$ for some i. Then we are done since this is an isomorphism of F-algebras and the K_i are mutually isomorphic F-algebras by the transitivity proved in Proposition 4.1. □

Theorem 4.3. $\#\,\mathrm{Gal}(K_i/F) = [K_i : F]$.

Proof. Note that $\mathrm{Gal}_i = \{\sigma \in S_n \mid \sigma(K_i) = K_i\}$ is the isotropy subgroup of K_i under the action of S_n on $\{K_1, \ldots, K_r\}$. Since this action is transitive by Proposition 4.1,

$$\#\,\mathrm{Gal}_i = \frac{n!}{r}.$$

However, we know that $K_1, \ldots, K_r$ are mutually isomorphic. Thus

$$n! = \dim_F(A) = [K_1 : F] + \cdots + [K_r : F] = r\,[K_i : F].$$

Combining this with the previous equation gives $\#\,\mathrm{Gal}_i = [K_i : F]$. Then we are done since $\mathrm{Gal}_i \simeq \mathrm{Gal}(K_i/F)$ by Proposition 4.1. □

Theorem 4.4. *If the characteristic of F doesn't divide $\#\,\mathrm{Gal}(K_i/F)$, then F is the fixed field of the action of $\mathrm{Gal}(K_i/F)$ on K_i.*

Proof. For brevity, we refer the reader to [4] for the proof. We should also note that a more general version of Theorem 4.4 is proved in [6]. □

4.3. Primary Decomposition

The Galois group of f consists of all permutations in S_n that preserve the algebraic structure of the roots. In this section, we will use primary decomposition to describe "the algebraic structure of the roots" and see how the Galois group "preserves" this structure. We will work over $\mathbb{Q}$ for simplicity.

Given $f = x^n - c_1x^{n-1} + \cdots + (-1)^n c_n \in \mathbb{Q}[x]$ as in Sect. 4.1, the splitting algebra is

$$A = \mathbb{Q}[x_1, \ldots, x_n]/\langle \sigma_1 - c_1, \ldots, \sigma_n - c_n\rangle.$$

Pick $h = t_1x_1 + \cdots + t_nx_n$ such that $m_h : A \to A$ is non-derogatory and let

$$\mathrm{CharPoly}_{m_h}(u) = \prod_{i=1}^{r} G_i(u)$$

be the irreducible factorization of the characteristic polynomial in $\mathbb{Q}[u]$. Similar to what we did in Sect. 3, the isomorphism (4.4) gives the primary decomposition

$$\langle \sigma_1 - c_1, \ldots, \sigma_n - c_n\rangle = \bigcap_{i=1}^{r} I_i$$

where

$$I_i = \langle \sigma_1 - c_1, \ldots, \sigma_n - c_n, G_i(h)\rangle.$$

4.3.1. Computing the Galois Group.

Note that S_n permutes the ideals I_i since $\langle \sigma_1 - c_1, \ldots, \sigma_n - c_n\rangle$ is invariant. It follows easily that

$$\mathrm{Gal}(K_i/\mathbb{Q}) \simeq \{\sigma \in S_n \mid \sigma(I_i) = I_i\}.$$

Using a Gröbner basis of I_i, we can determine whether $\sigma(I_i)$ equals I_i for any given $\sigma \in S_n$. Hence, by going through the elements of S_n one-by-one, we get a (horribly inefficient) algorithm for computing the Galois group.

4.3.2. Relations Among the Roots. Each ideal I_i in the primary decomposition is larger than $\langle\sigma_1 - c_1,\dots,\sigma_n - c_n\rangle$. The ideal $\langle\sigma_1 - c_1,\dots,\sigma_n - c_n\rangle$ encodes the obvious relations among the roots, and the polynomials we add to get from $\langle\sigma_1 - c_1,\dots,\sigma_n - c_n\rangle$ to I_i reflect the extra algebraic relations between the roots that hold in the splitting field K_i. Having more relations among the roots means that I_i is larger and hence K_i and the Galois group are smaller.

For instance, if the Galois group of f is S_n, then $\langle\sigma_1 - c_1,\dots,\sigma_n - c_n\rangle$ is a maximal ideal and the splitting algebra is the splitting field. This means that the *only* relations among the roots are the obvious ones relating the coefficients to the roots via the elementary symmetric polynomials.

We conclude with some examples that indicate what happens when the Galois group is smaller than S_n.

Example. Let $f = x^3 - c_1x^2 + c_2x - c_3 \in \mathbb{Q}[x]$ be an irreducible cubic. The splitting algebra of f is $A = \mathbb{Q}[x_1,x_2,x_3]/\langle\sigma_1 - c_1,\sigma_2 - c_2,\sigma_3 - c_3\rangle$. It is well-known that

$$\text{the Galois group of } f \text{ is isomorphic to } \begin{cases} S_3 & \text{if } \Delta(f) \notin \mathbb{Q}^2,\\ \mathbb{Z}/3\mathbb{Z} & \text{if } \Delta(f) \in \mathbb{Q}^2,\end{cases}$$

where $\Delta(f) \in \mathbb{Q}$ is the discriminant of f. By the above analysis, it follows that A is the splitting field of f when $\Delta(f) \notin \mathbb{Q}^2$.

Now suppose that $\Delta(f) = a^2$ for some $a \in \mathbb{Q}$. In this case, the splitting algebra is a product of two copies of the splitting field, i.e., $A = K_1 \times K_2$. Let

$$\sqrt{\Delta} = (x_1 - x_2)(x_1 - x_3)(x_2 - x_3) \in \mathbb{Q}[x_1,x_2,x_3].$$

In the splitting algebra A, we have $[\sqrt{\Delta}]^2 = [\Delta(f)]$, so that

$$[\sqrt{\Delta}]^2 = [a]^2.$$

Since A is not an integral domain, this does not imply $[\sqrt{\Delta}] = \pm[a]$. In fact, $[\sqrt{\Delta}] \in A$ cannot have a numerical value since $[\sqrt{\Delta}]$ is not invariant under S_3. Yet once we map to a field, the value must be $\pm a$. But which sign do we choose? The answer is *both*, which explains why we need two fields in the splitting algebra.

In this case, we have the primary decomposition

$$\langle\sigma_1 - c_1,\sigma_2 - c_2,\sigma_3 - c_3\rangle = I_1 \cap I_2,$$

where

$$\begin{aligned} I_1 &= \langle\sigma_1 - c_1,\sigma_2 - c_2,\sigma_3 - c_3,\sqrt{\Delta} - a\rangle,\\ I_2 &= \langle\sigma_1 - c_1,\sigma_2 - c_2,\sigma_3 - c_3,\sqrt{\Delta} + a\rangle.\end{aligned}$$

This primary decomposition show that the Galois group is A_3 since A_3 is the subgroup of S_3 that fixes $\sqrt{\Delta}$ and hence fixes I_1 and I_2.

The quartic is more complicated since there are five possibilities for the Galois group of an irreducible quartic. We will discuss the following case.

Example. Let $f = x^4 - c_1x^3 + c_2x^2 - c_3x + c_4 \in \mathbb{Q}[x]$ be an irreducible quartic with roots $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and splitting algebra

$$A = \mathbb{Q}[x_1, x_2, x_3, x_4]/\langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4\rangle.$$

To find the roots of f, we use the *Ferrari resolvent*

$$x^3 - c_2x^2 + (c_1c_3 - 4c_4)x - c_3^2 - c_1^2c_4 + 4c_2c_4. \tag{4.5}$$

Euler showed that if $\beta_1, \beta_2, \beta_3$ are the roots of (4.5), then the roots of f are

$$\frac{1}{4}\Big(c_1 \pm \sqrt{\beta_1 + c_1^2 - 4c_2} \pm \sqrt{\beta_2 + c_1^2 - 4c_2} \pm \sqrt{\beta_3 + c_1^2 - 4c_2}\Big),$$

where the signs are chosen so that the product of the square roots is $c_1^3 - 4c_1c_2 + 8c_3$. Also, as shown by Lagrange, the roots of the resolvent (4.5) are

$$\alpha_1\alpha_2 + \alpha_3\alpha_4,\ \alpha_1\alpha_3 + \alpha_2\alpha_4,\ \alpha_1\alpha_4 + \alpha_2\alpha_3. \tag{4.6}$$

The Galois group G of f over $\mathbb{Q}$ is isomorphic to one of the groups

$$S_4,\ A_4,\ D_8,\ \mathbb{Z}/4\mathbb{Z},\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z},$$

where D_8 is the dihedral group of order 8. Three cases are easy to distinguish:

$$G \simeq \begin{cases} S_4 & \text{if } \Delta(f) \notin \mathbb{Q}^2 \text{ and (4.5) is irreducible over } \mathbb{Q}, \\ A_4 & \text{if } \Delta(f) \in \mathbb{Q}^2 \text{ and (4.5) is irreducible over } \mathbb{Q}, \\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} & \text{if } \Delta(f) \in \mathbb{Q}^2 \text{ and (4.5) is reducible over } \mathbb{Q}. \end{cases}$$

The remaining case is when $\Delta(f) \notin \mathbb{Q}^2$ and (4.5) has a root in $\mathbb{Q}$. Here, the Galois group is D_8 or $\mathbb{Z}/4\mathbb{Z}$. We state without proof the following nice fact:

$$\begin{aligned} G \simeq D_8 \iff \ & \Delta(f) \notin \mathbb{Q}^2,\ \text{(4.5) has a root } b \in \mathbb{Q}, \text{ and} \\ & \quad \langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4\rangle = I_1 \cap I_2 \cap I_3 \\ & \text{is the primary decomposition, where} \\ & \quad I_1 = \langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4, x_1x_2 + x_3x_4 - b\rangle, \\ & \quad I_2 = \langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4, x_1x_3 + x_2x_4 - b\rangle, \\ & \quad I_3 = \langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4, x_1x_4 + x_2x_3 - b\rangle. \end{aligned}$$

The reason for three ideals is that b is one of the three combinations of roots given in (4.6). To get a field out of the ideal $\langle \sigma_1 - c_1, \sigma_2 - c_2, \sigma_3 - c_3, \sigma_4 - c_4\rangle$, we must commit to which combination gives b. This gives the ideals I_1, I_2, I_3 as above.

4.4. Numerical Methods

The earlier part of this section used symbolic methods to highlight the ideas of Galois theory. But when we turn our attention to computing Galois groups in practice, we encounter some interesting symbolic-numeric methods.

Most algorithms for computing Galois groups in computer algebra systems such as Maple, Magma and **GAP** use *resolvents.* For example, an irreducible quartic over $\mathbb{Q}$ has the Ferrari resolvent (4.5), whose factorization over $\mathbb{Q}$ helps determine the Galois group of the polynomial.

A more substantial example concerns the polynomial $f = x^7 - 154x + 99$ studied by Erbach, Fischer and McKay [8] in 1979. Let $\mathbb{F}_7$ be the finite field with 7 elements. The proof that the Galois group of f is isomorphic to $\mathrm{PSL}(2, \mathbb{F}_7)$ uses the resolvent R of degree $\binom{7}{3} = 35$ whose roots are all possible sums of three roots of f. The coefficients of R can be expressed in terms of the coefficents of f, though doing so symbolically would be very cumbersome. The paper [8] finds it more efficient to compute the roots of f numerically. This gives numerical approximations of the roots of R, from which R can be computed without difficultly, provided one pays sufficient attention to the accuracy. After this numerical computation, R is factored by symbolic methods, and the factors give crucial information about the Galois group of f.

A general introduction to the use of resolvents in computational Galois theory can be found in Chapter 14 of [3]. Some of the numerical issues are discussed in the paper [13], and a recent survey of computing Galois groups appears in [9]. Further references can be found at the end of Chapter 14 of [3].

References

[1] P. Aubry and A. Valibouze, *Using Galois ideals for computing relative resolvents*, J. Symbolic Comput. **30** (2000), 635–651.

[2] W. Auzinger and H.J. Stetter, *An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations*, in *Numerical Mathematics, Singapore 1988* (R.P. Agarwal, Y.M. Chow, and S.J. Wilson, editors), International Series of Numerical Mathematics **86**, Birkhäuser, Basel, 1988, 11–30.

[3] D.A. Cox, *Galois Theory*, John Wiley & Sons, Hoboken, 2004.

[4] D.A. Cox, *Solving equations via algebras*, in *Solving Polynomial Equations: Foundations, Algorithms, and Applications* (A. Dickenstein and I.Z. Emiris, editors), Springer-Verlag, New York Berlin Heidelberg, 2005, 63–123.

[5] H.M. Edwards, *Essays in Constructive Mathematics*, Springer-Verlag, New York Berlin Heidelberg, 2005.

[6] T. Ekedahl and D. Laksov, *Splitting algebras, symmetric functions, and Galois theory*, J. Algebra Appl. **4** (2005), 59–75.

[7] M. Elkadi and B. Mourrain, *Symbolic-numeric methods for solving polynomial equations and applications*, in *Solving Polynomial Equations: Foundations, Algorithms, and Applications* (A. Dickenstein and I.Z. Emiris, editors), Springer-Verlag, New York Berlin Heidelberg, 2005, 125–168.

[8] D.W. Erbach, J. Fischer and J. McKay, *Polynomials with $PSL(2, 7)$ as Galois group*, J. Number Theory **11** (1979), 69–75.

[9] A. Hulpke, *Techniques for computation of Galois groups*, in *Algorithmic Algebra and Number Theory, Heidelberg 1997* (B. Matzat, G.-M. Greuel, and G. Hiss, editors), Springer-Verlag, New York Berlin Heidelberg, 1999, 65–77.

[10] L. Kronecker, *Leopold Kronecker's Werke*, Volumes II and III, B.G. Teubner, Leipzig, 1897 (Volume II) and 1899/1931 (Volume III). Reprint by Chelsea, New York, 1968.

[11] D. Lazard, *Résolutions des systémes d'equations algébriques*, Theor. Comp. Sci. **15** (1981), 77–110.

[12] M. Pohst and H. Zassenhaus, *Algorithmic Number Theory*, Cambridge Univ. Press, Cambridge, 1989.

[13] R.P. Stauduhar, *The determination of Galois groups*, Math. Comp. **27** (1973), 981–996.

[14] H.J. Stetter, *Numerical Polynomial Algebra*, SIAM, Philadelphia, 2004.

David A. Cox
Department of Mathematics and Computer Science
Amherst College
Amherst, MA 01002 USA
e-mail: `dac@cs.amherst.edu`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 223–243
© 2007 Birkhäuser Verlag Basel/Switzerland

Pythagore's Dilemma, Symbolic-Numeric Computation, and the Border Basis Method

Bernard Mourrain

Abstract. In this tutorial paper, we first discuss the motivation of doing symbolic-numeric computation, with the aim of developing efficient and certified polynomial solvers. We give a quick overview of fundamental algebraic properties, used to recover the roots of a polynomial system, when we know the multiplicative structure of its quotient algebra. Then, we describe the border basis method, justifying and illustrating the approach on several simple examples. In particular, we show its usefulness in the context of solving polynomial systems, with approximate coefficients. The main results are recalled and we prove a new result on the syzygies, naturally associated with commutation properties. Finally, we describe an algorithm and its implementation for computing such border bases.

1. Introduction

Polynomial system solving is ubiquitous in many applications such as geometric modeling, robotics, computer vision, computational biology, and signal processing.

In CAGD (Computer-Aided Geometric Design) for instance, the objects of a scene or a piece to be built are represented by piecewise-algebraic models (such as spline functions or NURBS), which are able to encode the geometry of an object in a compact way. Indeed this B-spline representations is heavily used in CAGD, being now a standard for the representation of shapes. From a practical point of view, critical operations such as computing intersection curves of parameterized surfaces, or analyzing their topology are performed on these geometric models, which require, in fine, to solve polynomial equations.

In robotics or molecular biology (rebuilding of a molecule starting from the matrix of the distances between its atoms obtained by NMR), we have to compute positions of solids, satisfying polynomial equations, deduced from distance constraints. In signal processing, computing high-order statistics from signal observations leads to polynomial equations on the parameters that we want to identify.

Typical methods like minimization techniques and Newton-like methods are often used in these problems, but they do not always offer guarantees. They are local methods which do not provide global information on the set of solutions of the problems. However in many applications, it is important to detect all the possible solutions (usually all real solutions in a given domain).

In this paper, we give a tutorial presentation of a symbolic-numeric method for solving a polynomial system $f_1 = \cdots = f_m = 0$, which yields such global information on the roots.

Our objective is to devise certified and output-sensitive methods, in order to combine control and efficiency. How can we realize this objective?

First we have to make the context of our computation precise. The numbers that we can encode on a computer are integers, floating point numbers with fixed size. Such arithmetic is the basis of all symbolic and numeric methods in scientific computation. But there exist also dedicated efficient libraries to compute with integers, rational numbers or floating numbers of arbitrary size. Whereas in numerical computation, one uses fixed size floating point arithmetic, in symbolic computation, one is inclined to use large integer or rational numbers. But we should be aware that these number types, which are the basements of our computation, cannot represent all the numbers that we need in our modeling problems.

This is not a new problem. A long time ago, in the Ancient Greeks works, Geometry, the art of measuring the world, was already closely tied to arithmetic problems. Pythagore developed a complete model of computation, relating geometric constructions to (commensurable) numbers that we call today rational numbers. But Hyppase de Metaponte exhibited publically some weakness of this model (namely that $\sqrt{2}$ is not a rational number). The story says that this act of bravery had terrible consequences for him. Today, we want to deal with models of the real world on a computer. But this machine is able to compute efficiently only with fixed size or floating point numbers and we are facing again Pythagore's dilemma:

- *Should we consider that floating point arithmetic is sufficient to analyze all these problems?*
- *Should we accept to deal systematically with exact but implicit representation?*

The roots of symbolic-numeric computation can, somehow, be found in these questions. On one side, symbolic or exact computation allows to answer in a certified way, to many geometric questions such as counting the number of real roots in a domain, but suffers from a swell of complexity involving huge implicit representations. On the other hand, numerical computation is usually very efficient in approximating locally a given solution, but lacks for a global view on all the possible solutions. The objective of symbolic-numeric computation is to combine efficiency, doing approximate combination and certification, controlling the errors from the symbolic models.

The two main challenges, in this context, are to devise methods

- *which are numerically stable, when we perturb slightly the input coefficients,*
- *and which allow to control and improve the approximation level.*

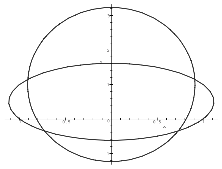
The first point is a prerequisite in the treatment of polynomial systems, for which the coefficients are known with some uncertainty. Such situations appear in many domains, such as CAGD, robotics, signal processing, where the models are not exact, due to measurement errors or to rounding operations, during the geometric processing steps.

Here is an example, which illustrates an instability in the algorithmic approach, which is not an instability of the problem. Consider a system of equations in two variables of the form

$$\begin{cases} p_1 := a\,x_1^2 + b\,x_2^2 + l_1(x_1, x_2) = 0, \\ p_2 := c\,x_1^2 + d\,x_2^2 + l_2(x_1, x_2) = 0, \end{cases}$$

where $a, b, c, d \in \mathbb{C}$ are complex numbers, $a\,d - b\,c \neq 0$ and l_1, l_2 are linear forms.

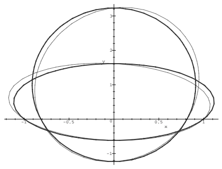
Let us compute a Gröbner basis [6] of these polynomials for a monomial order refining the degree order. The initial ideal is generated by (x_1^2, x_2^2) and the corresponding basis of $\mathfrak{A} = \mathbb{C}[x_1, x_2]/(p_1, p_2)$ is $\{1, x_1, x_2, x_1\,x_2\}$.



The two conics have horizontal and vertical axes and the basis $(1, x_1, x_2, x_1\,x_2)$ of $\mathfrak{A} = \mathbb{C}[x_1, x_2]/(p_1, p_2)$ is easily deduced from the Gröbner basis computation. Consider now a small perturbation of this system

$$\begin{cases} \tilde{p}_1 = p_1 + \epsilon_1\,x_1\,x_2, \\ \tilde{p}_2 = p_2 + \epsilon_2\,x_1\,x_2, \end{cases}$$

where $\epsilon_1, \epsilon_2 \in \mathbb{C}$ are "small" parameters. The zero-set is also the points of intersection of the two conics, which are slightly deformed but the initial ideal is now $(x_1^2, x_1 x_2, x_2^3)$ (if $x_1 \succ x_2$).



The basis of $\mathfrak{A} = \mathbb{C}[x_1, x_2]/(\tilde{p}_1, \tilde{p}_2)$, deduced from a Gröbner basis computation for the same monomial ordering, becomes $\{1, x_1, x_2, x_2^2\}$.

We see that, in the result of a small perturbation, basis may "jump" from one set of monomials to another, though the two situations are very closed to each other from a geometric point of view. Moreover, some of the polynomials of the Gröbner basis have large coefficients, for we have to divide by coefficients of the order ϵ_1, ϵ_2. This computation illustrates an unstable algorithm in a geometrically stable configuration.

Let us illustrate also the need to analyze approximation levels and to study convergence problems on the classical Wilkinson polynomial of degree 20:

$$p = \prod_{i=1}^{20} (x - i).$$

Expanding this product using large integer, and solving p with an appropriate solver (here we use `mpsolve` developed by D. Bini and G. Fiorentino [3]) yields the roots $1, 2, \ldots, 20$. Consider now the polynomial

$$q := p + 10^{-19}\, x^{19}.$$

If we solve with the same solver, we obtain the following approximations of the roots:

$$\left\{\begin{array}{l}
0.999999999999885647030 + \mathbf{i} * 0.780304076295825340952\, 10^{-30},\\
1.99999999991869592542 - \mathbf{i} * 0.355997149308138207546\, 10^{-25},\\
3.00000163363722549548 - \mathbf{i} * 0.193933463965337673953\, 10^{-21},\\
3.99783995916073831012 - \mathbf{i} * 0.149038769767887986466\, 10^{-12}\\
4.92749594405462332247 \pm \mathbf{i} * 0.362154009244065822206,\\
5.46579204715263866632 \pm \mathbf{i} * 1.42160717840964156977,\\
6.02565971634962238568 \pm \mathbf{i} * 2.81462226694663675275,\\
6.67233216337412127217 \pm \mathbf{i} * 4.70875887169693640999,\\
7.57026709806031661287 \pm \mathbf{i} * 7.48404858744277845517,\\
9.34247692903625548411 \pm \mathbf{i} * 12.0445559069247813966,\\
15.3308398638862630747 \pm \mathbf{i} * 20.5636861821969070263,\\
44.1662154433454716695 \pm \mathbf{i} * 24.4655059912717440795.
\end{array}\right.$$

Since the roots of the polynomial p are simple, applying the implicit function theorem, we can show that locally, they are continuous functions of the coefficients of the polynomial. In other words, for any $\epsilon > 0$, there exists a $\delta > 0$ such that a small perturbation of the input coefficients of at most δ induces a perturbation of at most ϵ on the roots. From a geometric point of view, we are in a stable situation.

However, the perturbed example shows that if we want a perturbation of order, say, 10^{-3} on the roots, we should consider approximation of the input coefficients at a precision much less than 10^{-19}. Here, a perturbation of 10^{-19} on the coefficients of p induces a perturbation of size > 20 on some of the roots.

A main challenge in symbolic-numeric methods is thus to analyze how the approximation on the input and output are related, and how to improve efficiently this level of approximation. In this tutorial paper, we are not going to elaborate on this difficult problem, but focus on the stability question in algebraic solvers.

2. From the Structure of $\mathfrak{A}$ to the Roots

In this section, we recall classical results, useful for solving effectively a polynomial system. We illustrate these on small examples.

We next define some notations which we will use hereafter. Let $\mathbb{K}$ be an effective field. The ring of n-variate polynomials over $\mathbb{K}$ will be denoted by R, $R = \mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \dots, x_n]$. We consider n-variate polynomials $f_1, \dots, f_s \in R$. Our goal is to solve the system of equations $f_1 = 0, \dots, f_s = 0$ over the algebraic closure $\overline{\mathbb{K}}$ of $\mathbb{K}$. These polynomials generate an ideal of $\mathbb{K}[\mathbf{x}]$ that we call I. From now on, we suppose that I *is zero-dimensional* so that its number of roots over $\overline{\mathbb{K}}$ is finite. The set of roots, with coordinates in the algebraic closure of $\mathbb{K}$, will be denoted by $\mathcal{Z}_{\overline{\mathbb{K}}^n}(I) = \{\zeta_1, \dots, \zeta_d\}$, with $\zeta_i = (\zeta_{i,1}, \dots, \zeta_{i,n}) \in \overline{\mathbb{K}}^n$.

2.1. The Quotient Algebra

We denote by $\mathfrak{A} = R/I$ the quotient algebra of R by I, that is the set of classes of polynomials in R modulo the ideal I. The class of an element $p \in R$, is denoted by $\overline{p} \in \mathfrak{A}$. Equality in $\mathfrak{A}$ is denoted by $\equiv$ and we have $a \equiv a'$ iff $a - a' \in I$.

The hypothesis that $\mathcal{Z}(I)$ is finite implies that the $\mathbb{K}$-vector space $\mathfrak{A}$ is of finite dimension (say D) over $\mathbb{K}$ [6, 8]. As we will see, we will transform the resolution of the non-linear system $\mathbf{f} = 0$, into linear algebra problems in the vector space $\mathfrak{A}$, which exploits its algebraic structure. Let us start with an example of computation in the quotient ring $\mathfrak{A}$.

Example 2.1. Let I be the ideal of $R = \mathbb{K}[x_1, x_2]$ generated by

$$\begin{aligned} f_1 &= 13\,x_1^2 + 8\,x_1\,x_2 + 4\,x_2^2 - 8\,x_1 - 8\,x_2 + 2, \\ f_2 &= x_1^2 + x_1\,x_2 - x_1 - \frac{1}{6}. \end{aligned}$$

The quotient ring $\mathfrak{A} = \mathbb{K}[x_1, x_2]/I$ is a vector space of dimension 4. A basis of $\mathfrak{A}$ is $1, x_1, x_2, x_1x_2$. We check that we have

$$\begin{aligned} x_1^2 &\equiv x_1^2 - f_2 = -x_1x_2 + x_1 + \frac{1}{6}, \\ x_1^2x_2 &\equiv x_1^2x_2 + \frac{1}{9}\,x_1\,f_1 - \left(\frac{5}{9} + \frac{13}{9}\,x_1 + \frac{4}{9}\,x_2\right) f_2 = -x_1x_2 + \frac{55}{54}\,x_1 + \frac{2}{27}\,x_2 + \frac{5}{54}. \end{aligned}$$

More generally, any polynomial in $\mathbb{K}[x_1, x_2]$ can be reduced, modulo the polynomials f_1, f_2, to a linear combination of the monomials $1, x_1, x_2, x_1\,x_2$, which as we will see form a basis of $\mathfrak{A}$.

Hereafter, $(\mathbf{x}^\alpha)_{\alpha \in E} = \mathbf{x}^E$ will denote a monomial basis of $\mathfrak{A}$. Any polynomial can be reduced modulo the polynomials $f_1, \dots, f_s$, to a linear combination of the monomials of the basis $\mathbf{x}^E$ of $\mathfrak{A}$.

2.2. The Dual

An important ingredient of our methods is the dual space $\widehat{R}$ that is, the space of linear forms $\Lambda : R \to \mathbb{K}$. The *evaluation at a point* $\zeta \in \mathbb{K}^n$ is a well-known example of such linear forms: $\mathbf{1}_\zeta : R \to \mathbb{K}$ such that $\forall p \in R$, $\mathbf{1}_\zeta(p) = p(\zeta)$.

Another class of linear forms is obtained by using differential operators. Namely, for any $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$, consider the map

$$\begin{array}{rcl} \mathbf{d}^\alpha : R & \to & \mathbb{K} \\ p & \mapsto & \dfrac{1}{\prod_{i=1}^n \alpha_i!} \, (d_{x_1})^{\alpha_1} \cdots (d_{x_n})^{\alpha_n} \, (p)(0), \end{array} \tag{1}$$

where d_{x_i} is the derivative with respect to the variable x_i. For a moment, we assume that $\mathbb{K}$ is of characteristic 0. We denote this linear form $\mathbf{d}^\alpha = (\mathbf{d}_1)^{\alpha_1} \cdots (\mathbf{d}_n)^{\alpha_n}$ and for any $(\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n, (\beta_1, \dots, \beta_n) \in \mathbb{N}^n$ observe that

$$\mathbf{d}^\alpha \left(\prod_{i=1}^n x_i^{\beta_i} \right) (0) = \left\{ \begin{array}{ll} 1 & \text{if } \forall i, \alpha_i = \beta_i, \\ 0 & \text{otherwise.} \end{array} \right.$$

It immediately follows that $(\mathbf{d}^\alpha)_{\alpha \in \mathbb{N}^n}$ is the dual basis of the primal monomial basis $(\mathbf{x}^\alpha)_{\alpha \in \mathbb{N}^n}$. Notice that $(\mathbf{d}^\alpha)_{\alpha \in \mathbb{N}^n}$ can be defined even in characteristic $\neq 0$. Hereafter, we will assume again that $\mathbb{K}$ is a field of arbitrary characteristic. By applying Taylor's expansion formula at 0, we decompose any linear form $\Lambda \in \widehat{R}$ as $\Lambda = \sum_{\alpha \in \mathbb{N}^n} \Lambda(\mathbf{x}^\alpha) \, \mathbf{d}^\alpha$. In particular, the evaluation $\mathbf{1}_\zeta$ is represented by

$$\mathbf{1}_\zeta = \sum_\alpha \zeta^\alpha \mathbf{d}^\alpha.$$

The map $\Lambda \to \sum_{\alpha \in \mathbb{N}^n} \Lambda(\mathbf{x}^\alpha) \, \mathbf{d}^\alpha$ defines a one-to-one correspondence between the set of linear forms Λ and the set $\mathbb{K}[[\mathbf{d}_1, \dots \mathbf{d}_n]] = \mathbb{K}[[\mathbf{d}]] = \{\sum_{\alpha \in \mathbb{N}^n} \Lambda_\alpha \, \mathbf{d}_1^{\alpha_1} \cdots \mathbf{d}_n^{\alpha_n}\}$ of formal power series (*f.p.s.*) in the variables $\mathbf{d}_1, \dots, \mathbf{d}_n$.

Hereafter, we will identify $\widehat{R}$ with $\mathbb{K}[[\mathbf{d}_1, \dots, \mathbf{d}_n]]$. The evaluation at 0 corresponds to the constant 1, under this definition. It will also be denoted $\mathbf{1}_0 = \mathbf{d}^0$.

Let us next examine the structure of the dual space. We can multiply a linear form by a polynomial ($\widehat{R}$ is an R-module) as follows. For any $p \in R$ and $\Lambda \in \widehat{R}$, we define $p \cdot \Lambda$ as the map $p \cdot \Lambda : R \to \mathbb{K}$ such that $\forall q \in R, p \cdot \Lambda(q) = \Lambda(p\,q)$. For any pair of elements $p \in R$ and for $\alpha_i \in \mathbb{N},\ \alpha_i \geq 1$, we check that we have $\mathbf{d}_i^{\alpha_i}(x_i\, p)(0) = d_i^{\alpha_i - 1} p(0)$. Consequently, for any pair of elements $p \in R, \alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$, where $\alpha_i \neq 0$ for a fixed i, we obtain that

$$x_i \cdot \mathbf{d}^\alpha(p) = \mathbf{d}^\alpha(x_i\, p) = \mathbf{d}_1^{\alpha_1} \cdots \mathbf{d}_{i-1}^{\alpha_{i-1}} \mathbf{d}_i^{\alpha_i - 1} \mathbf{d}_{i+1}^{\alpha_{i+1}} \cdots \mathbf{d}_n^{\alpha_n}(p),$$

that is, x_i acts as the *inverse* of $\mathbf{d}_i$ in $\mathbb{K}[[\mathbf{d}]]$. This is the reason why in the literature such a representation is referred to as the *inverse system* (see, for instance, [13, 16, 9]).

2.3. The Multiplication Operators

The first operator that comes naturally in the study of $\mathfrak{A}$ is the operator of multiplication by an element of $a \in \mathfrak{A}$. For any element $a \in \mathfrak{A}$, we define the map

$$\begin{array}{rcl} M_a : \mathfrak{A} & \to & \mathfrak{A} \\ b & \mapsto & a\, b. \end{array}$$

We will also consider the transposed operator

$$\begin{array}{rcl} M_a^t : \widehat{\mathfrak{A}} & \to & \widehat{\mathfrak{A}} \\ \Lambda & \mapsto & M_a^{\mathbf{t}}(\Lambda) = \Lambda \circ M_a. \end{array}$$

The matrix associated to this operator in the dual basis of a basis of $\mathfrak{A}$ is the transposed of the matrix of M_a in this basis.

Example 2.2. Let us compute the matrix of multiplication by x_1 in the basis $(1, x_1, x_2,\ x_1x_2)$ of $\mathfrak{A} = \mathbb{K}[x_1, x_2]/(f_1, f_2)$, where f_1, f_2 are the polynomials of Example 2.1. We multiply these monomials by x_1 and reduce them to a normal form. According to the computations of Example 2.1, we have

$$\begin{array}{rcl} 1 \times x_1 & \equiv & x_1, \\ x_1 \times x_1 & \equiv & -x_1x_2 + x_1 + \dfrac{1}{6}, \\ x_2 \times x_1 & \equiv & x_1x_2, \\ x_1x_2 \times x_1 & \equiv & -x_1x_2 + \dfrac{55}{54}x_1 + \dfrac{2}{27}x_2 + \dfrac{5}{54}, \end{array}$$

so that we have

$$M_1 = \begin{bmatrix} 0 & \frac{1}{6} & 0 & \frac{5}{54} \\ 1 & 1 & 0 & \frac{55}{54} \\ 0 & 0 & 0 & \frac{2}{27} \\ 0 & -1 & 1 & -1 \end{bmatrix}.$$

The multiplication map can be computed, when a normal form algorithm is available. In the next section, we will describe how to compute such a normal form, in a symbolic-numeric setting.

The algebraic solver approach is based on the following fundamental theorem (see [2, 14, 23]):

Theorem 2.3. *Assume that $\mathcal{Z}_{\overline{\mathbb{K}}^n}(I) = \{\zeta_1, \ldots, \zeta_d\}$.*

1. *The eigenvalues of the linear operator M_a (resp. $M_a^{\mathbf{t}}$) are $\{a(\zeta_1), \ldots, a(\zeta_d)\}$.*
2. *The common eigenvectors of $(M_a^{\mathbf{t}})_{a\in\mathfrak{A}}$ are (up to a scalar) $\mathbf{1}_{\zeta_1}, \ldots, \mathbf{1}_{\zeta_d}$.*

Notice that if $(\mathbf{x}^\alpha)_{\alpha\in E}$ is a monomial basis of $\mathfrak{A}$, then the coordinates of the evaluation $\mathbf{1}_{\zeta_i}$ in the dual basis of $(\mathbf{x}^\alpha)_{\alpha\in E}$ are $(\zeta_i^\alpha)_{\alpha\in E}$ where $\zeta^\alpha = \mathbf{1}_\zeta(\mathbf{x}^\alpha)$. Thus, if the basis $(\mathbf{x}^\alpha)_{\alpha\in E}$ contains $1, x_1, \ldots, x_n$ (which is often the case), the coordinates $[v_\alpha]_{\alpha\in E}$ (in the dual basis) of the eigenvectors of $M_a^{\mathbf{t}}$ yield all the coordinates of the root: $\zeta = [\frac{v_{x_1}}{v_1}, \ldots, \frac{v_{x_n}}{v_1}]$. This leads to the following algorithm:

Algorithm 2.4. Solving in the case of simple roots.

Let $a \in R$ and $\mathtt{M}_a$ be the matrix of multiplication in a basis $\mathbf{x}^E = (1, x_1, \ldots, x_n, \ldots)$ of $\mathfrak{A}$.

1. Compute the eigenvectors $\Lambda = [\Lambda_1, \Lambda_{x_1}, \ldots, \Lambda_{x_n}, \ldots]$ of $\mathtt{M}_a^{\mathbf{t}}$.
2. For each eigenvector Λ with $\Lambda_1 \neq 0$, compute and output $\zeta = \left(\frac{\Lambda_{x_1}}{\Lambda_1}, \ldots, \frac{\Lambda_{x_n}}{\Lambda_1}\right)$.

The set of output points ζ contains the set of simple roots of $\mathcal{Z}(I)$, since for such roots the eigenspace is one-dimensional. But as we will see on the next example, it can also yield in some cases[1] the multiple roots:

Example 2.1 *continued.* We compute the eigenvalues, their multiplicity, and the corresponding normalized eigenvector of the transposed of the matrix of multiplication by x_1:

Eigenvector	Eigenvalue	Multiplicity
$[1, -\frac{1}{3}, \frac{5}{6}, -\frac{5}{18}]$	$-\frac{1}{3}$	2
$[1, \frac{1}{3}, \frac{7}{6}, \frac{7}{18}]$	$\frac{1}{3}$	2

As the basis chosen for the computation is $(1, x_1, x_2, x_1x_2)$, the previous theorem tells us that the solutions of the system can be read off, from the 2nd and the 3rd coordinates of the normalized eigenvectors:

$$\zeta_1 = (-\frac{1}{3}, \frac{5}{6}) \text{ and } \zeta_2 = (\frac{1}{3}, \frac{7}{6}).$$

Moreover, the 4th coordinate of these vectors is the product of the 2nd by the 3rd coordinates.

In order to compute exactly the set of roots, counted with their multiplicity, we employ the following theorem. It is based on the fact that commuting matrices share common eigenspaces.

Theorem 2.5 ([14, 16, 5])**.** *There exists a basis of* $\mathfrak{A}$ *such that* $\forall a \in R$*, the matrix* M_a *is, in this basis, of the form*

$$\mathtt{M}_a = \begin{bmatrix} \mathtt{N}_a^1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathtt{N}_a^d \end{bmatrix} \text{ with } \mathtt{N}_a^i = \begin{bmatrix} a(\zeta_i) & & \star \\ & \ddots & \\ \mathbf{0} & & a(\zeta_i) \end{bmatrix}.$$

Here again, it leads to an algorithm:

[1]depending on the type of multiplicity

Algorithm 2.6. Solving by simultaneous triangulation.

```
INPUT: The matrices of multiplication $\mathtt{M}_{x_i}$ $(i = 1, \ldots, n)$ in a basis of $\mathfrak{A}$.
  1. Compute a (Schur) decomposition $\mathtt{P}$ such that all matrices $\mathtt{T}_{x_i} = \mathtt{PM}_{x_i}\mathtt{P}^{-1}$
     $(i = 1, \ldots, n)$ are upper-triangular.
  2. Compute the diagonal vectors $\mathbf{t}_i = (t^1_{i,i}, \ldots, t^n_{i,i})$ of the triangular matrices
     $\mathtt{T}_{x_i} = (t^i_{i,k})$ (for $i = 1, \ldots, D$).
OUTPUT: $\mathbf{t}_i, i = 0, \ldots, D$ the solutions of the input polynomial system, repeated
with their multiplicity.
```

The first step is performed by computing an ordered Schur decomposition of M_l (where l is a generic linear form) which yields a matrix $\mathtt{P}$ of change of basis. Next, we compute the matrices $\mathtt{T}_{x_i} = \mathtt{PM}_{x_i}\mathtt{P}^{-1}$ $(i = 1, \ldots, n)$ which are triangular, since they commute with M_l. The decomposition of the multiplication operators in Theorem 2.5 is in fact induced by a decomposition of the algebra

$$\mathfrak{A} = \mathfrak{A}_1 \oplus \cdots \oplus \mathfrak{A}_d,$$

where $\mathfrak{A}_i$ is the local algebra associated with the root ζ_i. More precisely, there exist elements $\mathbf{e}_1, \ldots, \mathbf{e}_d \in \mathfrak{A}$, such that for $i, j = 1, \ldots, d$,

$$\begin{cases} \mathbf{e}_i^2 \equiv \mathbf{e}_i, \\ \mathbf{e}_i \mathbf{e}_j \equiv 0, \ i \neq j, \\ \mathbf{e}_1 + \cdots + \mathbf{e}_d \equiv 1. \end{cases}$$

These polynomials, which generalize the univariate Lagrange polynomials, are called the fundamental idempotents of $\mathfrak{A}$. They are such that $\mathfrak{A}_i = \mathbf{e}_i \mathfrak{A}$ and $\mathbf{e}_i(\zeta_j) = 1$ if $i = j$ and 0 otherwise. The dimension of the $\mathbb{K}$-vector space $\mathfrak{A}_i$ is the *multiplicity* μ_{ζ_i} of ζ_i. See [26, 14, 8].

2.4. An Exact Representation of the Roots

In some problems, it is important to have an exact representation of the roots, with which we can effectively compute. Hereafter, we recall how to represent them as the image, by a rational map, of the roots of a univariate polynomial. The Chow form of $\mathfrak{A}$ is the homogeneous polynomial in $\mathbf{u} = (u_0, \ldots, u_n)$ of degree D, defined by

$$\mathcal{C}_I(\mathbf{u}) = \det(u_0 + u_1\,\mathtt{M}_{x_1} + \cdots + u_n\,\mathtt{M}_{x_n}).$$

According to Theorem 2.5, we have:

Theorem 2.7. *The Chow form of* $\mathfrak{A}$ *is*

$$\mathcal{C}_I(\mathbf{u}) = \prod_{\zeta \in \mathcal{Z}(I)} (u_0 + u_1\zeta_1 + \cdots + u_n\zeta_n)^{\mu_\zeta},$$

where μ_ζ *is the multiplicity of* ζ.

Example 2.1 *continued.* We compute the Chow form of the variety $I = (f_1, f_2)$, using the matrices of multiplication by x_1 and x_2, computed previously. Then

$$\det(u_0 + u_1\, M_1 + u_2\, M_2) = \left(u_0 + \frac{1}{3}\, u_1 + \frac{7}{6}\, u_2\right)^2 \left(u_0 - \frac{1}{3}\, u_1 + \frac{5}{6}\, u_2\right)^2 .$$

We check that it is a product of linear forms, whose coefficients yield the roots $\zeta_1 = (-\frac{1}{3}, \frac{5}{6})$ and $\zeta_2 = (\frac{1}{3}, \frac{7}{6})$. The exponents yield the multiplicity of the roots (here 2).

From this Chow form, it is possible to deduce a rational representation of the points of $\mathcal{Z}(I)$, as describe in the following algorithm. See [21, 1, 22, 12] for more details.

Algorithm 2.8. Univariate Rational Representation.

```
INPUT: A multiple $\Delta(\mathbf{u})$ of the Chow form $I \subset R$.
  1. Compute the square-free part $d(u)$ of $\Delta(u)$.
  2. Choose a generic $\mathbf{t} \in \mathbb{K}^{n+1}$ and compute the first terms of
         $d(\mathbf{t} + \mathbf{u}) = d_0(u_0) + u_1\, d_1(u_0) + \cdots + u_n\, d_n(u_0) + \cdots$ .
  3. Compute the redundant rational representation $\zeta_1 = \frac{d_1(u_0)}{d_0'(u_0)}, \ldots, \zeta_n = \frac{d_n(u_0)}{d_0'(u_0)}$,
     $d_0(u_0) = 0$.
  4. Factor $d_0(u_0)$, keep the good prime factors and output the corresponding
     simplified rational univariate representations of the roots $\mathcal{Z}(I)$.
```

This result describes the coordinates of the roots of the polynomial system $f_1 = 0, \ldots, f_s = 0$, as the image by an explicit rational map of some of the roots of $d_0(u_0)$. Since we start with a multiple of $\Delta(\mathbf{u})$, in order to have exactly the roots we can remove the redundant factor of d_0, by substituting the rational representation back into the equations $f_1, \ldots, f_n$.

This completes our description of the quotient algebra $\mathfrak{A} = \mathbb{K}[\mathbf{x}]/I$, and shows how knowing its multiplicative structure yields a representation of the roots of I. We are now going to consider how to compute effectively the multiplication structure of $\mathfrak{A}$. For this purpose, we will consider normal form methods, which given a polynomial $p \in \mathbb{K}[\mathbf{x}]$, compute a canonical element for its class in $\mathfrak{A} = \mathbb{K}[\mathbf{x}]/I$ (or modulo I).

3. Border Basis Method

Gröbner basis computation yields, by reduction, the normal form of any element modulo I. As shown in the introduction, however by computing a Gröbner basis on a perturbed system, we obtain a completely different representation of $\mathfrak{A}$. In this section, we are going to detail an alternative approach, known as the border basis method, or generalized normal form method [15, 18, 25, 24, 19, 11, 10], and adapted to symbolic-numeric computation.

Example 3.1. Consider the system

$$f_1 := x_1^2 + x_2^2 - x_1 + x_2 - 2, \quad f_2 := x_1^2 - x_2^2 + 2\,x_2 - 3.$$

The computation of a Gröbner basis for the degree-lexicographic ordering yields

$$2\,{x_2}^2 - x_1 - x_2 + 1,\ 2\,{x_1}^2 - x_1 + 3\,x_2 - 5.$$

The leading monomials are x_1^2, x_2^2 and the corresponding monomial basis of $\mathfrak{A}$ is $\{1, x_1, x_2, x_1\,x_2\}$. Consider now a small perturbation:

$$f_1, f_2 + 10^{-7}\,x_1\,x_2.$$

We obtain

$$\begin{aligned}
2\,{x_2}^2 &\equiv +x_1 + x_2 - 1 + 0.0000001\,x_1x_2,\\
{x_1}^2 &\equiv -{x_2}^2 + \ x_1 - x_2 + 2,\\
x_1\,x_2 &\equiv 10000000.9999999999999950000000000000125\,{x_2}^2\\
&\quad -5000000.250000012499999374999968750001562500078125\,x_1\\
&\quad -5000000.750000037499993124999906250017187500234375\,x_2\\
&\quad +5000000.250000062499999374999843750001562500390625.
\end{aligned}$$

The leading monomials are now $x_1\,x_2, x_1^2, x_2^3$ and the corresponding basis of $\mathfrak{A}$ is $B = \{1, x_1, x_2, x_2^2\}$.

Notice however that $\{1, x_1, x_2, x_1x_2\}$ is still a basis of $\mathfrak{A}$ and that we have the following equivalences in $\mathfrak{A}$:

$$\begin{aligned}
x_1^2 &\equiv -0.00000005\,x_1x_2 + \tfrac{1}{2}\,x_1 - \tfrac{3}{2}\,x_2 + \tfrac{5}{2},\\
x_2^2 &\equiv +0.00000005\,x_1x_2 + \tfrac{1}{2}\,x_1 + \tfrac{1}{2}\,x_2 - \tfrac{1}{2},\\
x_2x_1^2 &\equiv 0.49999999\,x_1x_2 - 0.74999998\,x_1 + 1.75000003\,x_2 + 0.74999994,\\
x_1x_2^2 &\equiv 0.49999999\,x_1x_2 - 0.25000004\,x_1 - 0.74999991\,x_2 + 1.25000004.
\end{aligned}$$

In this representation, we observe that the perturbation on the latter rewriting rules is of the same order as on the input coefficients. We also notice that this set of linear relations between the monomials on the border of B and B yields directly the matrices of multiplication by x_1, x_2 in $\mathfrak{A}$, since we are able to compute the product of any element of the basis $B = \{1, x_1, x_2, x_1x_2\}$ by x_1 or x_2.

The key observations, that we deduce from this example, are the following:

- *If we are in a geometrically stable situation, a basis of the quotient algebra by a polynomial system will remain a basis for a small perturbation of this system.*
- *To compute the structure of $\mathfrak{A}$, we only have to know to which combination of the basis monomials, the monomials on the* border *are equivalent to.*

We are going to detail now, how these remarks can be turned into an algorithm. For more details on the method that we describe, see [15, 17, 25, 18, 19].

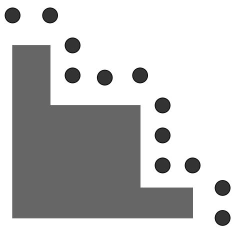
3.1. Border Monomials

The support supp(p) of a polynomial $p \in \mathbb{K}[\mathbf{x}]$ is the set of monomials appearing with non-zero coefficients in p. Given a set S of elements of $\mathbb{K}[\mathbf{x}]$, we denote by $\langle S \rangle$ the $\mathbb{K}$-vector space spanned by the elements of S. We denote the set of all the monomials in the variables $\mathbf{x} = (x_1, \ldots, x_n)$ by $\mathcal{M}$. The usual degree of a monomial $m \in \mathcal{M}$ will be denoted by $|m|$.

A set of monomials B is said to be *connected to* 1 if and only if, for every monomial m in B, there exists a finite sequence of variables $(x_{i_j})_{j\in[1,l]}$ such that $1 \in B$, $\Pi_{j=1\ldots l'} x_{i_j} \in B$, $\forall l' \in [1, l]$ and $\Pi_{j\in[1,l]}\, x_{i_j} \ = m$. A set of monomials B is said to be *stable by division* if $m = m'm''$ in B implies that m' and m'' are in B. Notice that if B is stable by division, it is connected to 1.

For any subset S of R, we denote by S^+ the set $S^+ = S \cup x_1\, S \cup \cdots \cup x_n\, S$, $\partial S = S^+ \backslash S$. If S is a set of monomials, ∂S with be called the set of *border monomials* of S.

In the following example, with two variables, B is the set of monomials under the staircase. It is stable by division. The set ∂B is formed by the dotted monomials.



In order to compute the structure of the quotient algebra $\mathfrak{A}$, we have

- to compute a (monomial) basis B of $\mathfrak{A}$,
- and for each border monomial m in ∂B, to compute the linear combination of monomials in B equal to it modulo the ideal I.

From this construction, we deduce directly the tables of multiplication by the variables and thus the roots, as described in Sect. 2.

3.2. Reduction

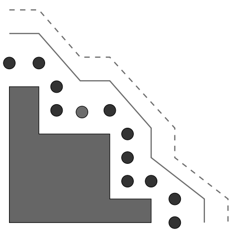
Suppose that we have a monomial set B containing 1, which might be a basis of $\mathfrak{A}$, and for each monomial $m \in \partial B$ we have computed $b_m \in \langle B \rangle$ such that $\rho_m = m - b_m \in I$. Let $F = (\rho_m)_{\partial B}$. F is called *a rewriting family* for B, if it has the following property: $\forall f, f' \in F$,

- f has exactly **one** monomial that we denoted by $\gamma(f)$ (also called the *leading monomial* of f) in ∂B,
- $\mathrm{supp}(f) \subset B^+$, $\mathrm{supp}(f - \gamma(f)) \subset B$,
- if $\gamma(f) = \gamma(f')$ then $f = f'$.

For $B = \{1, x, y, xy\}$, the set of polynomials $F = \{x^2 - 1, y^2 - x_1, x^2y - x_1, y^2x - y\}$ is a reducing family of degree 3.

This notion of rewriting family can be refined by the degree, as described in [19].

Once we have a rewriting family, we have a way to project any monomial onto $\langle B\rangle$ modulo the ideal I. For this purpose, we introduce the notion of B-index: let $B^{[i]} = (\ldots(B^{+})\ldots)^{+}$ be the application of i times the operator $^{+}$ on B. Since B contains 1, for any monomial $m \in \mathcal{M}$, there exists k such that $m \in B^{[k]}$. We say that a monomial m is of B-index k if $m \in B^{[k]} - B^{[k-1]}$, and we denote it by $\delta_B(m)$. By convention, $B^{[0]}$ is B, which is the set of monomials of B-index 0. We denote by $B^{[<k]} = B^{[0]} \cup \cdots \cup B^{[k-1]}$. We represent here the monomials of B-index 1 (the points), those of B-index 2 (first polygon) and 3 (dashed polygon):



Using the polynomials in F, any monomial in $B^{[1]} = \partial B$ can be rewritten in $\langle B\rangle$. By induction, we prove that any monomial $B^{[k]}$ can be rewritten in $\langle B^{[<k]}\rangle$ modulo the ideal generated by F. Thus iterating this reduction, any polynomial of $\mathbb{K}[\mathbf{x}]$ can be projected modulo (F) onto $\langle B\rangle$. For details and algorithms, see [15, 19]. Here also, we can refine this reduction, with respect to the degree or any graduation.

Let us denote by R_F such a linear projection from $\mathbb{K}[\mathbf{x}]$ to $\langle B\rangle$, deduced from the rewriting family F for a set B connected to 1. By definition, we have the following properties:

$$\begin{array}{l} \forall m \in B, R_F(m) = m, \\ \forall m \in \partial B, R_F(m) = m - \rho_m, \end{array}$$

where $\rho_m \in F$ is the unique member of F, which monomial $\in \partial B$ is m. More generally, for $p \in \langle B^{+}\rangle$, we have

$$R_F(p) = p - \sum_{m \in \mathrm{supp}(p) \cap \partial B} \lambda_m \, \rho_m,$$

where λ_m is the coefficient of m in p.

This allows us to define the following operator:

$$\begin{array}{rcl} M_i : \langle B\rangle & \to & \langle B\rangle \\ b & \mapsto & R_F(x_i b). \end{array}$$

It has the following properties: $\forall m \in B$,

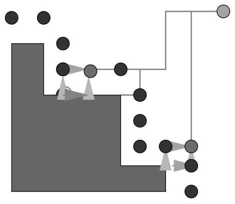
- if $x_i\, m \in B$, then $M_i(m) = x_i\, m$,
- otherwise $m' := x_i\, m \in \partial B$ is a border monomial, so that $M_i(m) = x_i m - \rho_{x_i\, m}$.

3.3. Normal Form Criterion

A question, which at this point is not solved, is how to check that B is really a basis of $\mathfrak{A}$. In other words, how can we check

- that $\mathbb{K}[\mathbf{x}] = \langle B \rangle \oplus (F)$, or equivalently
- that R_F is a normal form, modulo the ideal (F), or equivalently
- that the reduction by the rewriting family F is unique.

Let us illustrate the situation on a bivariate example:



In this picture, the isolated monomial can, a priori, be reduced in many different ways to a linear combination of elements in B. Each path down from this monomial to a monomial of B yields a distinct way to reduce it. In such a path, a horizontal step corresponds to the multiplication by x_1 and a vertical step to the multiplication by x_2. In order to ensure that the reduction is unique, we thus have to check that the multiplication by x_1 first, then the reduction R_F, then the multiplication by x_2 and again the reduction by R_F yield the same result than when we exchange the role of x_1 and x_2. In other words, we have to check that

$$M_1 \circ M_2 = M_2 \circ M_1$$

on B. It turns out that such commutation condition is sufficient to guaranty that we have a normal form, assuming that the basis B is connected to 1:

Theorem 3.2 ([15]). *Let $B \subset \mathcal{M}$ connected to 1. Let $R_F : B^+ \to B$ be a projection from B^+ to B, with kernel F and let $I = (F)$ be the ideal generated by F. Let*

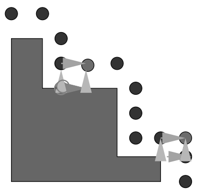
$$\begin{aligned} M_i : B &\to B \\ b &\mapsto R_F(x_i b). \end{aligned}$$

Then, the two properties are equivalent:

1. *For all $1 \le i, j \le n$, $M_i \circ M_j = M_j \circ M_i$.*
2. $\mathbb{K}[\mathbf{x}] = \langle B \rangle \oplus I$.

If this holds, the reduction induced by R_F is a normal form modulo I onto $\langle B \rangle$.

Effectively, it means that we have to check the commutation only for the monomials on the border of B, as it is discussed below and illustrated in this figure:



To simplify the presentation, let us assume now that B is *stable by division*. Let $m \in B$ and consider two variables x_i, x_j, with $i \neq j$.

- If $m' := x_i\, x_j\, m \in B$, then since B is stable by division, $x_i\, m, x_j\, m \in B$ and we have $M_i \circ M_j(m) = x_i\, x_j\, m = M_j \circ M_i(m)$.
- If $x_i\, m \in B$ but $x_j\, m \notin B$, we have

$$\begin{aligned} M_i(m) &= x_i\, m, \\ M_j(m) &= x_j\, m - \phi(x_j\, m), \\ M_j \circ M_i(m) &= x_j\, x_i\, m - \rho_{x_i\, x_j\, m}, \\ M_i \circ M_j(m) &= x_i M_j(m) + \sum_{\omega \in \partial B} \lambda_\omega \rho_\omega, \end{aligned}$$

so that $M_j \circ M_i(m) = M_i \circ M_j(m)$ implies that

$$\rho_{x_i\, m}\, x_i - \rho_{x_i\, x_j\, m} = \sum_{\omega \in \partial B} \lambda_\omega \rho_\omega. \tag{2}$$

- If $m\, x_i \notin B$ and $m\, x_j \notin B$, then

$$\begin{aligned} M_i(m) &= m\, x_i - \rho_{m\, x_i}, \\ M_j(m) &= m\, x_j - \rho_{m\, x_j}, \\ M_j \circ M_i(m) &= x_j M_i(m) - \sum_{\omega \in \partial B} \lambda_{j,\omega} \rho_\omega, \\ M_i \circ M_j(m) &= x_i M_j(m) - \sum_{\omega \in \partial B} \lambda_{j,\omega} \rho_\omega, \end{aligned}$$

and $M_j \circ M_i(m) = M_i \circ M_j(m)$ implies that

$$x_j \rho_{x_i\, m} - x_i \rho_{x_j\, m} = \sum_{\omega \in \partial B} (\lambda_{j,\omega} - \lambda_{i,\omega}) \rho_\omega. \tag{3}$$

The relations (2) and (3) are syzygies between the polynomials of F. They are called *next-door* and *across-the-street relations* in [10]. They are special forms of the C-polynomials (also called commutation polynomials in [19]) that we define as follows: For any polynomials $f_1, f_2 \in F$, let the C-polynomial relative to γ be

$$C(f_1, f_2) = \frac{\operatorname{lcm}(\gamma(f_1), \gamma(f_2))}{\gamma(f_1)} f_1 - \frac{\operatorname{lcm}(\gamma(f_1), \gamma(f_2))}{\gamma(f_2)} f_2.$$

Theorem 3.2 is equivalent to the following proposition:

Proposition 3.3 ([19]). *Assume that B is connected to 1 and that F is a rewriting family on B. If for all $f, f' \in F$ such that $C(f, f') \in B^+$, we have*

$$C(f, f') \in \langle F \rangle,$$

then B is a basis of $\mathfrak{A} = \mathbb{K}[\mathbf{x}]/(F)$.

This property generalizes the well-known property of the S-polynomials in the computation of Gröbner bases [4].

3.4. Syzygies and Commutation Relations

We are going to analyze more precisely the relations between the polynomials $F = (f_\omega)_{\omega \in \partial B}$. These relations or syzygies form a module that we denote by

$$\mathrm{Syz}(F) = \{\sum_\omega h_\omega e_\omega \in \mathbb{K}[\mathbf{x}]^{\partial B}; \sum_\omega h_\omega \rho_\omega = 0\},$$

where $(e_\omega)_{\omega \in \partial B}$ is the canonical basis of $\mathbb{K}[\mathbf{x}]^{\partial B}$.

We denote by Ξ the module of $\mathbb{K}[\mathbf{x}]^{\partial B}$ generated by the relations (2) and (3).

Lemma 3.4. *$\forall m \in \mathcal{M}$, $\forall \theta \in \partial B$,*

$$m\, e_\theta \equiv \sum_{\omega \in \partial B} m_\omega e_\omega \quad \text{modulo } \Xi,$$

with $\delta_B(m_\omega) = |m_\omega| + 1$.

Proof. By definition, we have $\theta \in \partial B$, so that $\delta_B(\theta) = 1$. If $\delta_B(m\,\theta) = |m| + 1$, the property is true. Otherwise, $\delta_B(m\,\theta) < |m| + 1$, which implies that there exists an $i_0 \in [1, \dots, n]$ such that $m = x_{i_0} m'$ with $x_{i_0}\theta \in \partial B$. Using the relations (2), we have

$$m\, e_\theta = m'\, x_{i_0} e_\theta \equiv m'(e_{x_{i_0}\theta} + \sum_{\omega \in \partial B} \lambda_\omega e_\omega),$$

with $|m'| < |m|$. By iterating this reduction (which will eventually stop), we prove that modulo the relations (2), we have $m\, e_\theta \equiv \sum_{\omega \in \partial B} m_\omega e_\omega$ with $\delta_B(m_\omega) = |m_\omega| + 1$. □

Lemma 3.5. *$\forall m, m' \in \mathcal{M}$, $\forall \theta, \theta' \in \partial B$ such that $m\theta = m'\theta'$, $\theta \neq \theta'$ and $\delta_B(m\,\theta) = |m| + 1 = k$, $\delta_B(m'\,\theta') = |m'| + 1 = k$, we have*

$$m\, e_\theta - m' e_{\theta'} \equiv \tilde{m}\, e_{\tilde{\theta}} - m' e_{\theta'} + \sum_\omega h_\omega\, e_\omega \quad \text{modulo } \Xi,$$

with $\delta_B(h_\omega\, \rho_\omega) < k$ and $|\mathrm{lcm}(\tilde{\theta}, \theta')| < |\mathrm{lcm}(\theta, \theta')|$.

Proof. Let $\nu = \mathrm{lcm}(\theta, \theta')/\theta$, $\nu = \mathrm{lcm}(\theta, \theta')/\theta'$. As $\theta \neq \theta'$, $|\nu| \neq 0$ or $|\nu'| \neq 0$. If $|\nu| = 0$, then $|\nu'| > 0$ and $\theta = \nu'\theta'$ so that $\delta_B(m'\theta') \leq |m'| - |\nu'| + 1$, which is a contradiction. Similarly, we cannot have $|\nu'| = 0$.

This implies that there exists an $i_0 \neq i_1$ such that x_{i_0} divides θ (and m') and such that x_{i_1} divides θ' (and $m = x_{i_1} t$). Let us consider $\tilde{\theta} \in \partial B$ such that $x_{i_1}\theta = x_{i_0}\tilde{\theta}$. Using the relations (3), we have

$$x_{i_1} e_\theta \equiv x_{i_0} e_{\tilde{\theta}} + \sum_{\omega \in \partial B} \lambda_\omega \, e_\omega.$$

Therefore, we deduce that modulo the relations (3),

$$m e_\theta - m' e_{\theta'} \equiv \tilde{m} e_{\tilde{\theta}} - m' e_{\theta'} + \sum_{\omega} \lambda_\omega \, t \, e_\omega,$$

with $\tilde{m} = x_{i_0} t$, $\delta_B(t e_\omega) \leq |t| + 1 < |m| + 1$ and $t x_{i_0} \theta = m' \theta'$, so that $|\mathrm{lcm}(\tilde{\theta}, \theta')| < |\mathrm{lcm}(\theta, \theta')|$. □

This yields the following, conjectured in [10]:

Theorem 3.6. *Assume that B is stable by division and that we have a rewriting family $F = (\rho_\omega)_{\omega \in \partial B}$, satisfying the conditions of Theorem* 3.2. *Then* $\mathrm{Syz}(F)$ *is generated by the relations* (2) *and* (3).

Proof. Let $\sigma = \sum_\omega p_\omega \, e_\omega \in \mathrm{Syz}(F)$. Applying Lemma 3.4, by reduction modulo Ξ, we may assume that $\delta_B(p_\omega \omega) = |p_\omega| + 1$. Let us consider the maximum of such B-indices.

As $\sum_\omega p_\omega \, \rho_\omega = 0$, there exist $\theta \neq \theta' \in \partial B$ and monomials $m \in \mathrm{supp}(p_\theta)$, $m' \in \mathrm{supp}(p_{\theta'})$ such that $m\,\theta = m'\,\theta'$. By Lemma 3.5, we can replace this pair $(m\, e_\theta, m'\, e_{\theta'})$ by a new pair where either the degree of $\mathrm{lcm}(\theta, \theta')$ or the B-index of $m\,\theta$ (resp. $m'\,\theta'$) is smaller.

Since we cannot iterate infinitely these reductions steps, we deduce that σ is in the module generated by the relations (2) and (3). □

4. Algorithm and Software

This analysis leads to the following scheme of algorithm, for computing a normal form, modulo the ideal generated by the polynomials $F = (f_1, \ldots, f_s)$.

Since we want to use rewriting rules on monomials, at some point of our computation, we will need to choose a specific monomial in a given polynomial. For that purpose, we introduce a choice function $\gamma : \mathbb{K}[\mathbf{x}] \to \mathcal{M}$ satisfying the following properties:

- $\gamma(p) \in \mathrm{supp}(p)$,
- if $m \in \mathrm{supp}(p)$, $m \neq \gamma(p)$ then $\gamma(p)$ does not divide m,
- and $\Lambda(\gamma(p)) = \max\{\Lambda(m),\ m \in \mathrm{supp}(p)\}$,

where Λ is a "degree" or a graduation of $\mathbb{K}[\mathbf{x}]$.

This choice function has some similarity with the leading term function in Gröbner basis constructions, but offers much more freedom. For instance, on can use the *Macaulay* choice function γ, such that for all $p \in \mathbb{K}[\mathbf{x}]$, $\gamma(p) = x_1^{\alpha_1} \cdots$

$x_n^{\alpha_n}$ satisfies $\deg_{\mathbb{N}}(\gamma(p)) = \max\{\deg_{\mathbb{N}}(m); m \in \mathrm{supp}(p)\} = d$, and $\exists i_0$ such that $\alpha_{i_0} = \max\{\deg_{x_i}(m), m \in \mathrm{supp}(p) \text{ and } \deg_{\mathbb{N}}(m) = d; i = 1, \dots, n\}$.

Or one can take, among the monomials of largest degree, the one with the coefficients of largest modulus, if we are working over $\mathbb{R}$ or $\mathbb{C}$.

Algorithm 4.1. Compute the generalized normal form.

```
INPUT: $f_1, \dots, f_s \in \mathbb{K}[\mathbf{x}]$ such that $I = (f_1, \dots, f_s)$ defines a complex variety of dimension 0.
  • Initialize $P$ with the polynomials of $F$ of minimal degree $k$ and $B$ with the
    set of monomials outside $\gamma(P)$.
  • Repeat
      1. Compute $\tilde{P}$ the set of polynomials of $P^+$ and of $C$-polynomials of $P$,
         which are in $B^+$.
      2. Apply linear transformation of the coefficient matrix of $P'$ in order to
         rewrite monomials of $\partial B$ in terms of the monomials in $B$.
      3. Update $B$,
           – either by removing the monomial multiples of $\gamma(p)$, for $p \in \langle \tilde{P} \rangle \cap \langle B \rangle$,
           – or by adding the monomials of $\partial B$, not in $\gamma(\tilde{P})$.
      4. Update $P$ by inserting the new polynomials obtained from step 2, with
         one monomial in $\partial B$ and the other in $B$.
    until $\partial B = \gamma(P)$.
OUTPUT: The rewriting family $P$ on the basis $B$ of $\mathfrak{A} = \mathbb{K}[\mathbf{x}]/I$.
```

The algorithm described here has been implemented by P. Trébuchet [25] in the library Synaps [2] (see `solve(L, Newmac<C>())`). It corresponds to about 50 000 lines of C++-code.

The main part is the computation of the linear algebra part, which consists in computing a triangular form of the coefficient matrix of the polynomials $\tilde{P}$. As this coefficient matrix may be sparse, it uses sparse LU decomposition algorithm, which avoid to produce too dense rows when performing column pivoting operations on the matrix. More precisely, the sparse matrix data structure `MatrSps<double, sparse::rep2d<double> >` based on the container `sparse::rep2d<T>` provides the vector space operations and the matrix-vector multiplication, for sparse matrices. The container type `sparse::rep2d<>` corresponds to the NCformat type of superlu [7], so that no copy of objects is needed to call the external routines. The LU decomposition routines of superlu have been ported in C++ to be able to use generic coefficients.

Once this triangular form is obtained, intersecting $\langle \tilde{P} \rangle$ and $\langle B \rangle$ consists just in extracting the rows of the matrix corresponding to polynomials with supports in $\langle B \rangle$.

[2]`http://www-sop.inria.fr/galaad/software/synaps/`

The numerical approximation of the roots are obtained by eigenvalues computation, using the library Lapack (the routine `dgegv`) and the strategy described in [5].

Several experiments have been performed to analyse the behavior of the methods, depending on the choice functions such as the choice function associated to the Degree Reverse Lexicographical order, or to the degree lexicographical order, a choice function that returns randomly any of the monomials of maximum degree of the polynomial given as its input, the Macaulay's choice function, the choice function over the rational that minimize the memory needed in the reduction loop, ... We also experiment with different types of coefficients such as rational numbers, or extended floating point numbers, to analyze the size and the precision of the computation. See [19] or [20], for detailed results.

It turns out that in many situations, the Macaulay choice function produces representation of the quotient algebra $\mathfrak{A}$, which is more compact than the others, faster to compute and which requires less accuracy. This is not always the case, and understanding how to compute the optimal representation of $\mathfrak{A}$ from a numerical and algebraic point of view is still an open and challenging problem.

5. Open Problems

This new approach for constructing border bases is a generalisation of Gröbner basis computation. Not tied to a monomial order, it provides more freedom to perform polynomial reductions. In linear algebra, matrix triangulation with column pivoting is much better, from a numerical point of view, than triangulation without column pivoting. Similarly in this new approach, we can choose pivots according to numerical criteria. However, many open problems remain to be solved:

- What is the optimal strategy to obtain a compact description of the quotient algebra, when dealing with exact coefficients?
- When we are computing with approximate coefficients, which reduction strategy should we adopt to obtain a minimal numerical error on the description of this quotient algebra?
- How to determine tuned thresholds in zero-tests, when performing the numerical matrix reductions?
- How can we estimate the condition number of the approximation of the quotient algebra?
- Can we connect this condition number with the error on the solutions?
- Is there a Newton-like way to improve efficiently the level of approximation of the quotient algebra?

As we said, the interaction between symbolic and numeric computations is a fascinating area, where many important problems are waiting for solutions.

References

[1] M.E. Alonso, E. Becker, M.F. Roy and T. Wörmann. Zeros, multiplicities and idempotents for zero-dimensional systems. In L. González-Vega and T. Recio, editors, *Algorithms in Algebraic Geometry and Applications*, volume 143 of *Prog. in Math.*, pages 1–15. Birkhäuser, Basel, 1996.

[2] W. Auzinger and H.J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Proc. Intern. Conf. on Numerical Math.*, volume 86 of *Int. Series of Numerical Math*, pages 12–30. Birkhäuser, Basel, 1988.

[3] D. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13, 1996.

[4] B. Buchberger. Gröbner bases: An algebraic method in ideal theory. In N. K. Bose, editor, *Multidimensional System Theory*, pages 184–232. Reidel Publishing Co., 1985.

[5] R.M. Corless, P.M. Gianni and B.M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In W. W. Küchlin, editor, *Proc. ISSAC '97*, pages 133–140. ACM Press, New York, 1997.

[6] D. Cox, J. Little and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1992.

[7] J. Demmel, J. Gilbert and X.S. Li. An asynchronous parallel supernodal algorithm for sparse gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 20(4): 915–952, 1999.

[8] M. Elkadi and B. Mourrain. *Introduction à la résolution des systèmes d'équations algébriques*, 2003. Notes de cours, Univ. de Nice (310 p.).

[9] P.A. Fuhrmann. *A Polynomial Approach to Linear Algebra.* Springer-Verlag, Berlin, 1996.

[10] A. Kehrein and M. Kreuzer. A characterisation of border bases. *J. Pure and Applied Algebra*, 196: 251–270, 2005.

[11] A. Kehrein, M. Kreuzer and L. Robbiano. An algebraist's view on border bases. In A. Dickenstein and I. Emiris, editors, *Solving Polynomial Equations: Foundations, Algorithms, and Applications*, volume 14 of *Algorithms and Computation in Mathematics*, pages 169–202. Springer-Verlag, Berlin, 2005.

[12] G. Lecerf. Computing an equidimensional decomposition of an algebraic varety bymeans of geometric resolutions. In *Proc. ISSAC 2000*, pages 209–216. ACM Press, New York, 2000.

[13] F.S. Macaulay. *The Algebraic Theory of Modular Systems.* Cambridge Univ. Press, Cambridge, 1916.

[14] B. Mourrain. Computing isolated polynomial roots by matrix methods. *J. Symbolic Computation, Special Issue on Symbolic-Numeric Algebra for Polynomials*, 26(6): 715–738, 1998.

[15] B. Mourrain. A new criterion for normal form algorithms. In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Proc. AAECC*, volume 1719 of *LNCS*, pages 430–443. Springer-Verlag, Berlin, 1999.

[16] B. Mourrain and V.Y. Pan. Multivariate polynomials, duality and structured matrices. *J. Complexity*, 16(1): 110–180, 2000.

[17] B. Mourrain and P. Trébuchet. Solving projective complete intersection faster. In C. Traverso, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 231–238. ACM Press, New York, 2000.

[18] B. Mourrain and P. Trébuchet. Algebraic methods for numerical solving. In *Proc. 3rd International Workshop on Symbolic and Numeric Algorithms for Scientific Computing '01* (Timisoara, Romania), pages 42–57, 2002.

[19] B. Mourrain and P. Trébuchet. Generalised normal forms and polynomial system solving. In M. Kauers, editor, *Proc. Intern. Symp. on Symbolic and Algebraic Computation*, pages 253–260. ACM Press, New York, 2005.

[20] B. Mourrain and P. Trébuchet. Generalised normal forms and polynomial system solving. Technical Report 5471, INRIA Sophia-Antipolis, 2005.

[21] J. Renegar. On the computational complexity and geometry of the first order theory of reals (I, II, III). *J. Symbolic Computation*, 13(3): 255–352, 1992.

[22] F. Rouillier. Solving zero-dimensional polynomial systems throuhg Rational Univariate Representation. *App. Alg. Eng. Com. Comp.*, 9(5): 433–461, 1999.

[23] H.J. Stetter. Eigenproblems are at the heart of polynomial system solving. *SIGSAM Bulletin*, 30(4): 22–25, 1996.

[24] H.J. Stetter. *Numerical Polynomial Algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004.

[25] P. Trébuchet. *Vers une résolution stable et rapide des équations algébriques*. PhD thesis, Université Pierre et Marie Curie, 2002.

[26] W.V. Vasconcelos. *Computational Methods in Commutative Algebra and Algebraic Geometry*, volume 2 of *Algorithms and Computation in Mathematics*. Springer-Verlag, New York, 1998.

Bernard Mourrain
GALAAD, INRIA
BP 93 06902 Sophia Antipolis France
e-mail: `mourrain@sophia.inria.fr`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 245–267
© 2007 Birkhäuser Verlag Basel/Switzerland

Proposal for the Algorithmic Use of the BKK-Number in the Algebraic Reduction of a 0-dimensional Polynomial System

Hans J. Stetter

Abstract. For a regular 0-dimensional system P of polynomials with numerical coefficients, its BKK-number m equals the number of its zeros, counting multiplicities. In this paper, I analyze how the knowledge of m may be used for the computation of a Gröbner basis or more generally a border basis of P. It is also shown how numerical stability may be preserved in such an approach, and how near-singular systems are recognized and handled. There remain a number of open questions which should stimulate further research.

1. Introduction

When I prepared my invited lecture for SNC 2005, I had no intention of publishing it. Therefore, for the second part of my talk, I chose to put forward some preliminary ideas which I felt were worth being investigated further. After the presentation of the lecture in Xi'an, I was urged by a number of colleagues to publish its content, which — with a good deal of hesitation — I finally agreed to do. A later more thorough consideration of such a publication convinced me that there should actually be two separate papers: one should contain the first part of my talk which explained facts about the mathematical feasibility of extending significant parts of polynomial algebra into the realm of approximate data and approximate computation. The other one should put down the ideas in the second part of my talk in a more elaborate and formal fashion, as a stimulus for further research. This is that second paper.

It contains my ideas about a novel algorithmic approach to the numerical computation of a standard representation of the ideal or of the quotient ring resp. of a *regular 0-dimensional* system of polynomials with *numerical coefficients*. Here "regular" is used in analogy with its usage in Numerical Linear Algebra: no tiny change of coefficients can change the dimension of the quotient ring; cf. [11, Sect. 8].

The approach is based on the fact that the *exact number of zeros* of such a system (counting multiplicities) or — equivalently — the *exact vector dimension* of the quotient ring can be determined *a priori* by a symbolic algorithm with only the *supports* of the individual polynomials as input. I have wondered for many years why, by my observation, nobody in the large GB community has attempted to simplify the GB computation for regular 0-dimensional systems P by using its BKK-number as a helpful input, particularly after BKK computation has become a standard tool in the late 1990s; cf. e.g. [16, 10].

In the following, I present the steps of an algorithm which — in a good number of cases — computes the elements of a *border basis* of the ideal generated by P or, equivalently, the set of *multiplication matrices* of the associated quotient ring, both w.r.t. a particular *normal set* or monomial basis. The algorithm uses *no term order*; therefore it requires various choices and decisions which may determine its success; I have only been able to indicate potentially successful strategies. Besides these choices, the complete computation of the final result (border basis, multiplication matrices) consists only of substitutions and of the solution of blocks of *linear* vector equations. In particular, *no reductions to zero* are needed, which is an important feature for the numerical computation.

I have no proof that the algorithm will always succeed, even for the most clever choices in the preparatory part. This paper is rather an invitation to refine its ideas into a true algorithm, if only for some particular subclasses of regular 0-dimensional systems, or to establish that it must fail almost always. In any case, a good deal of insight should result from these investigations (which I am too old to tackle).

The paper begins with a summary of facts about border bases and multiplication matrices; it also introduces my terminology and notation. In Sect. 3, we show how the *syzygies* of a border basis, or the *consistency conditions* for the associated multiplication matrices, may be reduced to a minimal set which should — together with the system P — specify the basis uniquely. All this refers to a particular normal set whose number of elements must equal the BKK-number of P; its selection is discussed in Sect. 4. Then the constituent steps of the proposed algorithm are introduced and discussed in Sect. 5. In the following Sect. 6, the potential appearance of *ill-conditioning* is considered and algorithmic remedies are explained. Sect. 7 then deals with the possibility that the specified regular system P is actually *very close to singularity*: one type of singularity (BKK deficiency) may be dealt with algorithmically, and the other type (positive dimension) can only be diagnosed. The following Sect. 8 discusses various details of a potential implementation and points out potential sources of failure. Some conclusions complete the paper.

It should be mentioned that many parts of the material in this paper are also contained somewhere in my book on “Numerical Polynomial Algebra” [14]; but this paper presents them in a self-contained and systematic fashion which should help in bringing them to the attention of the community. I am also aware that some ideas in this paper are related to ideas in the more recent GB-algorithms of J.-C. Faugère (cf., e.g. [3]) and to ideas in papers by B. Mourrain and P. Trébuchet (cf., e.g., [11, 12]); but since their approaches and their notational framework differ from mine considerably, I have not pointed this out in detail. A more refined development of the algorithm proposed in this paper is contained in a forthcoming paper [7] of A. Kehrein and M. Kreuzer. Also I wish to thank SIAM Publ. for granting the permission to use figures which are adaptations of figures in [14].

2. Border Bases and Multiplication Tables

We denote by $\mathbb{P}^s$ the set (ideal) of all polynomials in s variables with coefficients in $\mathbb{C}$. Monomials in $\mathbb{P}^s$ are denoted by $x^k := x_1^{k_1} \dots x_s^{k_s}$, $k \in \mathbb{N}_0^s$. A set $\mathcal{N}$ of monomials in $\mathbb{P}^s$ is *closed* if $x^\mu \in \mathcal{N}$ implies $x^{\mu'} \in \mathcal{N}$ for all divisors $x^{\mu'}$ of x^μ. In $\mathbb{P}^s$, we consider a 0-dimensional *polynomial ideal* $\mathcal{I}$ with m zeros (counting multiplicities), with its *quotient ring* $\mathcal{R}[\mathcal{I}] := \mathbb{P}^s / \mathcal{I}$ of vector space dimension m. A closed m-element monomial set $\mathcal{N}$ is a *feasible normal set* for $\mathcal{R}$ if it is a basis of $\mathcal{R}$ as a vector space.

A regular polynomial system $P \subset \mathbb{P}^s$, with s equations, generates a *0-dimensional* ideal whose quotient ring dimension m *equals* the BKK-number of P. Note that the BKK-number of P depends only on the *supports* of the s polynomials in P, and that it may be computed via the mixed volume of the Newton polytope of P; cf., e.g., [15, 10].

Definition 2.1. For a closed monomial set $\mathcal{N} \subset \mathbb{P}^s$, the set of all monomials which are *not* in $\mathcal{N}$ but satisfy $x^k = x_\sigma x^j$ for some $x^j \in \mathcal{N}$ and some $x_\sigma, \sigma = 1(1)s$, is the *border set* $B[\mathcal{N}]$ of $\mathcal{N}$, with elements x^{k_ν}, $\nu = 1(1)N := |B[\mathcal{N}]|$.

It is well-known that the *multiplicative structure* of $\mathcal{R}$ is specified, w.r.t. the fixed basis $\mathcal{N}$, by the *multiplication matrices* $A_\sigma \in \mathbb{C}^{m \times m}$, $\sigma = 1(1)s$, which represent the residue classes $\bmod \mathcal{I}$ of the elements in $B[\mathcal{N}]$ in terms of the basis $\mathcal{N}$, with the elements x^{j_μ} of $\mathcal{N}$ arranged into a vector $\mathbf{b}$ in some arbitrary but *fixed* order:

$$x_\sigma \mathbf{b} \equiv A_\sigma \mathbf{b} \bmod \mathcal{I}, \quad \sigma = 1(1)s. \tag{1}$$

While many of the rows in the A_σ are simply unit rows because they refer to a normal set element "inside" $\mathcal{N}$ with the x_σ-shifted monomial also in $\mathcal{N}$, there must exist a particular *nontrivial* row $a_{k_\nu}^T$ in A_σ for each element x^{j_μ} of $\mathcal{N}$ with an x_σ-neighbor x^{k_ν} in $B[\mathcal{N}]$: $x^{k_\nu} = x_\sigma x^{j_\mu} \equiv a_{k_\nu}^T \mathbf{b}$. If an element x^{k_ν} in $B[\mathcal{N}]$ is an x_σ-neighbor of a normal set monomial for several distinct σ, then the same row $a_{k_\nu}^T$ will occur in all of the respective matrices A_σ.

Obviously, the collection of these nontrivial rows specifies the multiplicative structure of $\mathcal{R}[\mathcal{I}]$ and thus also the structure of $\mathcal{I}$:

Definition 2.2. The set of the $N := |B[\mathcal{N}]|$ polynomials

$$bb_\nu(x) := x^{k_\nu} - a_{k_\nu}^T \mathbf{b}, \quad \nu = 1(1)N, \tag{2}$$

is a *border basis* $\mathcal{B}[\mathcal{I}]$ of $\mathcal{I}$; more specifically, it is the $\mathcal{N}$-*border basis* $\mathcal{B}_{\mathcal{N}[\mathcal{I}]}$ of $\mathcal{I}$. Cf. also [6, 8].

Except for $s = 1$, the number N of border basis elements bb_ν is $> s$, often $N \gg s$. Thus $\mathcal{B}_{\mathcal{N}[\mathcal{I}]}$ would be *overdetermined* if its elements bb_ν were considered as independent. $\mathcal{B}_{\mathcal{N}[\mathcal{I}]}$ can define a nontrivial ideal with m zeros *only* if the bb_ν satisfy the *system of syzygies* which arises in the following way:

In the border set $B[\mathcal{N}]$, two monomials x^{k_ν}, $x^{k_{\nu'}}$ are *neighbors* if they satisfy one of the following two relations:

$$\begin{aligned} &\text{(i) For some } \sigma, \quad & x^{k'_\nu} &= x_\sigma\, x^{k_\nu}\,; \\ &\text{(ii) For some } \sigma, \sigma', & x_\sigma\, x^{k'_\nu} &= x_{\sigma'}\, x^{k_\nu}\,. \end{aligned} \tag{3}$$

For the coefficients of the border basis elements bb_ν, $bb_{\nu'}$ associated with neighboring border monomials, this implies (cf. (1), (2), (3)) in case

$$\begin{aligned} \text{(i) } 0 &= \quad bb_{\nu'} - x_{\sigma'} bb_\nu + (a^T_{k_{\nu'}} - a^T_{k_\nu}\, x_{\sigma'})\,\mathbf{b} &&\equiv\; (a^T_{k_{\nu'}} - a^T_{k_\nu}\, A_{\sigma'})\,\mathbf{b} \in \mathcal{I}\,; \\ \text{(ii) } 0 &= x_\sigma bb_{\nu'} - x_{\sigma'} bb_\nu + (a^T_{k_{\nu'}}\, x_\sigma - a^T_{k_\nu}\, x_{\sigma'})\,\mathbf{b} &&\equiv\; (a^T_{k_{\nu'}}\, A_\sigma - a^T_{k_\nu}\, A_{\sigma'})\,\mathbf{b} \in \mathcal{I}\,. \end{aligned} \tag{4}$$

Note that $a^T\,\mathbf{b} \in \mathcal{I}$ implies $a = 0$. Therefore we have

Theorem 2.1. *For a specified feasible normal set* $\mathcal{N}$*, the row vectors* $a^T_{k_\nu} \in \mathbb{C}^m$, $\nu = 1(1)N$ *whose components are the coefficients of the border basis* $\mathcal{B}_\mathcal{N}$ *of the 0-dimensional polynomial ideal* $\mathcal{I}$ (*cf.* (2)) *and the elements of the nontrivial rows of the multiplication matrices* A_σ *w.r.t.* $\mathcal{N}$ *of the quotient ring* $\mathcal{R}[\mathcal{I}]$ (*cf.* (1)) *must satisfy, for neighboring border set monomials* (*cf.* (3))*:*

$$\begin{aligned} &\textit{in case (i)} \quad & a^T_{k_{\nu'}} &= a^T_{k_\nu}\, A_{\sigma'}\,; \\ &\textit{in case (ii)} & a^T_{k_{\nu'}}\, A_\sigma &= a^T_{k_\nu}\, A_{\sigma'}\,. \end{aligned} \tag{5}$$

Theorem 2.1 gives necessary relations for the border basis polynomials of $\mathcal{I}$ as well as for the multiplication matrices A_σ of $\mathcal{R}[\mathcal{I}]$. For the latter, *commutativity* is known to be a necessary and sufficient condition, cf. [11]. But a closer analysis shows (cf. [14], Thm. 8.11) that the relations (5) are *identical* with the relations arising from the commutativity conditions $A_{\sigma'}\, A_\sigma = A_\sigma\, A_{\sigma'}$ for all pairs (σ, σ') when these are spelt out in terms of the rows $a^T_{k_\nu}$.

Corollary 2.2. *The conditions* (5) *of Theorem* 2.1 *are necessary and sufficient.*

Example 2.1. In $\mathbb{P}^3$, we consider a 0-dimensional ideal with $m = 6$ zeros and assume that the 6-point normal set $\mathcal{N}$ shown in Fig. 1 is feasible. (The figure depicts a set of monomials x^{j_μ} in $\mathbb{P}^3$ by the set of its exponents j_μ in $\mathbb{N}^3$; this is a commonly used visualization tool.) The associated border set $B[\mathcal{N}]$ is also indicated; each of its monomials is a "positive neighbor" of a monomial in $\mathcal{N}$. The total number N of border monomials turns out to be 9. The edges in the figure connect border monomials which are *neighbors* in the sense defined above. There are $\bar{N} = 14$ pairs of neighbors, four of which are of type (i) while the others are of type (ii).

As an example of a relation (4) of type (i), we consider the neighboring monomials $x^{(0,1,1)} = x_2x_3$ and $x^{(1,1,1)} = x_1x_2x_3$: they require the validity of

$$(a^T_{(1,1,1)} - a^T_{(0,1,1)}\, x_1)\, \mathbf{b} \in \mathcal{I}\,. \tag{6}$$

To use this relation computationally, we must replace those monomials x^k of $x_1\mathbf{b}$ which are in $B[\mathcal{N}]$ by $a_k^T\mathbf{b}$. When we now assume that — within an algorithm for the computation of the $a^T_{k_\nu}$ — the row vector $a^T_{(0,1,1)}$ has already been found, then the above relation constitutes a *linear* vector equation for the remaining unknown vectors $a^T_{k_\nu}$.

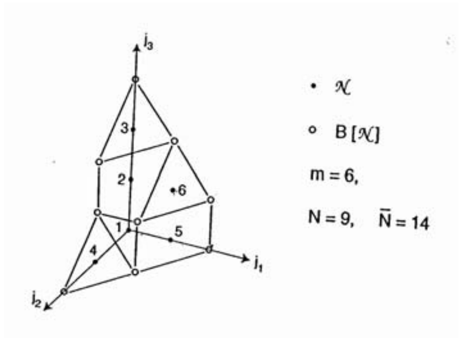


FIGURE 1

On the other hand, when we consider the type (ii) neighbors $x^{(1,1,1)} = x_1x_2x_3$ and $x^{(1,0,2)} = x_1x_3^2$ which require the validity of

$$(a^T_{(1,0,2)}\, x_2 - a^T_{(1,1,1)}\, x_3)\, \mathbf{b} \in \mathcal{I}\,,$$

and assume the row vector $a^T_{(1,1,1)}$ to be known, then we will obtain a linear vector equation only if $a^T_{(1,0,2)}$ is known, too.

3. Minimal Syzygy Bases

It is our goal to use the syzygy relations (4) for the computation of the nontrivial rows of the multiplication matrices for a specified feasible normal set, in the fashion indicated in Example 2.1. If we wanted to do this for an ideal generated by polynomials with integer or simple rational coefficients and in rational arithmetic, we might be willing to put up with the *overdetermination* which prevails in the complete system (4). For the *numerical treatment* of (4) which we have in mind, such an overdetermination is fatal: since the use of floating-point arithmetic excludes algorithmic decisions based on a strict equality of complex numbers, the interdependence of the relations (4) may remain undetected and lead to an *inconsistent* system.

At first, we consider the number of syzygy relations which should be operative in a minimal set. To determine N nontrivial vectors $a^T_{k_\nu}$ we should have N

independent vector equations. s of those will be furnished by the polynomial system which defines the ideal at hand. Thus, $N - s$ further independent equations should be necessary and sufficient to determine the $a_{k_\nu}^T$ uniquely. The following concept yields an overview of the complete set of relations (4):

Definition 3.1. For a specified normal set $\mathcal{N}$, the *border web* $BW_{\mathcal{N}}$ is the set of all pairs of monomials in $B[\mathcal{N}]$ which are neighbors in the sense of (3). In the visualization of monomials by their exponents, the border web $BW_{\mathcal{N}}$ is the set of all *edges* connecting the exponents of neighboring pairs. In this representation, $BW_{\mathcal{N}}$ is a *graph* in s-space.

This concept of a border web permits us to use graph theoretic considerations for the reduction of the system (4) to a subset of $N - s$ independent relations. At first, we realize that the graph $BW_{\mathcal{N}}$ contains *closed loops.* But a relation (4) represented by an edge "closing a loop" holds if the relations for the remaining edges of the loop hold; cf. [14, Propositions 8.14 and 8.15].

By a well-known theorem of graph theory , the "breaking of all loops" in a connected graph with N nodes results in a graph with exactly $N - 1$ edges. While this has brought us closer to our goal, we still have to get rid of $s - 1$ further relations or edges resp.

We note that our restriction to syzygies arising from neighboring border monomials is not compulsory. Like with Gröbner bases, syzygies between border basis polynomials whose leading terms are *relatively prime* are satisfied automatically; cf. [2]. In our visualization, relatively prime border monomials become nodes in the border web which lie in disjoint subspaces (e.g. $x^{(1,0,0)}$ and $x^{(0,0,3)}$). Therefore we are permitted to augment our border web graph by *"virtual"* edges between such nodes.

The introduction of these further edges makes it possible to delete some further "real" edges of $BW_{\mathcal{N}}$ which close a loop with a virtual edge. Altogether, $s-1$ real edges may be deleted because $s-1$ virtual edges may be introduced without creating loops between virtual edges; cf. [14, Proposition 8.14 (b)]. This reduces the total number of remaining edges and thus of operative equations (4) to the desired $N-s$. Such a *minimal border web* will be denoted by $\overline{BW}_{\mathcal{N}}$. Naturally, throughout the deletion process, we must take care that the graph remains connected and that each one of the final support leading monomials of the autoreduced system retains some *real* edge(s) issuing from it.

Example 3.1. We continue with the situation of Example 2.1; cf. the visualized border web of Fig. 1. When we consider the node $(2, 0, 0)$ as the root of a tree, it becomes obvious that the 3 edges in the 2, 3-plane close loops. After their deletion, we may still delete the two upward edges issuing form $(1, 1, 0)$ and the edge between $(1, 1, 1)$ and $(1, 0, 2)$. This leaves $N - 1 = 8$ edges.

Now we introduce the two virtual edges from $(2, 0, 0)$ to $(0, 2, 0)$ and $(0, 0, 3)$ resp. which are obviously in the disjoint subspaces 1-axis, 2-axis, 3-axis; cf. Fig. 2. This enables us to delete the real edges ending in $(0, 2, 0)$ and $(0, 0, 3)$ without

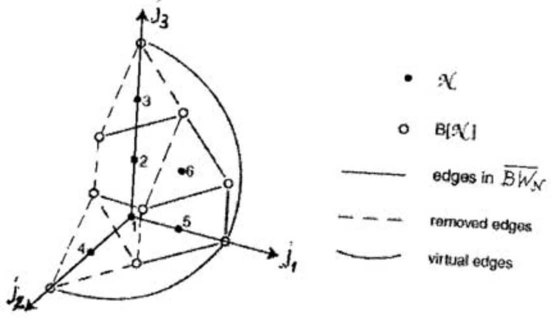


FIGURE 2

disconnecting these nodes from the graph. Thus, we have arrived at $N - s = 6$ conditions (4) which, together with the 3 generating polynomials of $\mathcal{I}$, should suffice to specify the complete border basis.

The minimal border web $BW_\mathcal{N}$ is not at all uniquely determined. E.g., we could have retained the type (i) edges $[(1,1,0),(1,1,1)]$ and $[(0,1,1),(0,1,2)]$ and deleted the type (ii) edges $[(2,0,1),(1,1,1)]$ and $[(1,0,2),(0,1,2)]$ instead.

Actually, in the algorithmic computation of a border basis, the minimal border web is not selected *a priori* but formed recursively during the computation; cf. Sects. 5 and 8.

4. Selection of a Tentative Normal Set

As explained in the Introduction, we assume that we *know* the number m of zeros (counting multiplicities) of the 0-dimensional ideal $\mathcal{I}$ defined by the *regular* polynomial system $P = \{p_\nu,\ \nu = 1(1)s\}$. This number is also the expected vector space dimension of the quotient ring $\mathcal{R}[\mathcal{I}]$ and hence the number of elements in a monomial basis $\mathcal{N}$ of $\mathcal{R}$. Remember that the number m computed from the mixed volume of the Newton polytope of P (cf. e.g. [4]) depends only on the *supports* of the polynomials p_ν and not on their specified coefficients. Thus, a closed set $\mathcal{N}$ with the correct number m of monomials and a structure compatible with the supports of the p_ν may still not be *feasible* for the actual coefficients of P. This can only be discovered during the border basis computation; the appropriate measures will be discussed in later sections.

As a first step towards the selection of a normal set for $\mathcal{R}$ we attempt to simplify the system P by *autoreduction*:

In Gröbner basis computation, autoreduction is controlled by term order: we check whether the leading monomial of some p_ν divides terms in another polynomial $p_{\nu'}$. If yes, we eliminate these terms, in the sequence of term order, by the subtraction of suitable multiples of p_ν. The possibly new leading monomial of the reduced $p_{\nu'}$ may now divide terms in another polynomial etc. Clearly, this simplification procedure does not change the ideal $\mathcal{I}[P]$. It must stop because term orders can only decrease in this well-known procedure.

Border bases do not refer to a term order but to a normal set $\mathcal{N}$, with a partial ordering given *a-posteriori* by the "distance" to $\mathcal{N}$; cf. [14, Definition 8.3]. But autoreduction must take place *before* the specification of the normal set. Therefore, we proceed as follows:

We consider the monomial sets S_ν of the *supports* of the p_ν and their *internal borders* consisting of those monomials in the S_ν with no multiples in S_ν. For each p_ν, we select a monomial in the internal border of its support S_ν as *support-leading* monomial which we use like the (term order)-leading monomial above. In the absence of term order, we must avoid eliminations in a $p_{\nu'}$ which would introduce new border terms into $S_{\nu'}$; this requires the *simultaneous* elimination in sets of polynomials where the support-leading terms of each polynomial also occur in the other polynomials. This is easily achieved by the solution of a linear system for the support-leading terms. Naturally, each elimination in a $p_{\nu'}$ redefines $S_{\nu'}$ and may require the selection of a new support-leading monomial which — possibly — permits further elimination. Because the supports can only shrink, the procedure must come to a stop. However, depending on the choice of the support-leading monomials, the results may differ considerably; cf. Example 4.1 below. The support sets of the final autoreduced system, *without* the respective support-leading monomials, will be denoted by $\bar{S}_\nu$.

Now we are ready to select a normal set $\mathcal{N}$ of proper magnitude m. For this purpose we take the *union* of the s truncated final support sets $\bar{S}_\nu$. Then we complement this union into a *closed convex* monomial set, i.e. a set which contains all divisors of one of its elements. If the resulting set has exactly m elements, it is a (tentative) normal set for $\mathcal{R}[\mathcal{I}]$.

Otherwise, we have to adjoin further monomials such that the set remains convex and that no multiples of the final support-leading monomials are appended. For an arbitrary choice of support-leading monomials, it may not be possible to reach a set of m elements. But we know that m member normal sets for $\mathcal{R}[\mathcal{I}]$ must exist, with one monomial of each autoreduced support not in $\mathcal{N}$; therefore it must be possible to reach such a normal set, perhaps with some additional sophistication in the approach.

Example 4.1. To avoid confusing complications, we take a very simple situation: $s = 2$, and dense polynomials p_1, p_2 with degrees 2 and 3 resp.; this makes $m = 6$. A less trivial situation will appear in Sect. 7; cf. Example 7.2 and Fig. 5 (i).

(i) At first, we follow the Gröbner basis pattern. Use of the term order `tdeg`$[x_2, x_1]$ makes x_2^2 and x_2^3 the leading monomials in p_1, p_2 resp. With suitable multiples of p_1, we may eliminate the x_2^3, $x_1x_2^2$ and x_2^2 terms in p_2; the leading monomial of the reduced p_2 is $x_1^2x_2$, and the union of the truncated supports is

$\{1,\, x_1,\, x_2,\, x_1^2,\, x_1x_2,\, x_1^3\}$ (cf. Fig. 3); it has exactly 6 elements. This normal set also arises in the course of a Gröbner basis algorithm; but its *a priori* knowledge cuts the basis computation short; cf. Example 5.1 (i).

(ii) Without a term order, we may choose the monomials x_2^2 and x_1^3 as support-leading monomials of p_1, p_2 resp.; note that the internal border of the two supports consists of the monomials of degree 2 and 3 resp. The reduction proceeds as before; but now the support-leading monomial of p_2 remains unchanged throughout; the union $\{1,\, x_1,\, x_2,\, x_1^2,\, x_1x_2,\, x_1^2x_2\}$ of the final truncated supports has $m = 6$ elements and is a nice normal set for a border basis of our $\{p_1,\, p_2\}$; cf. Example 5.1 (ii).

(iii) Another normal set arises when we take x_1x_2 as the support-leading monomial of p_1, which we are free to do. Now we can eliminate the $x_1x_2^2$ and $x_1^2x_2$ terms in p_2 and choose either x_2^3 or x_1^3 as support leading monomial in the reduced p_2. For the former choice, we obtain $\{1,\, x_1,\, x_2,\, x_1^2,\, x_2^2,\, x_1^3\}$ as normal set for a border basis. It looks a little strange but functions alright as we shall see in Example 5.1 (iii).

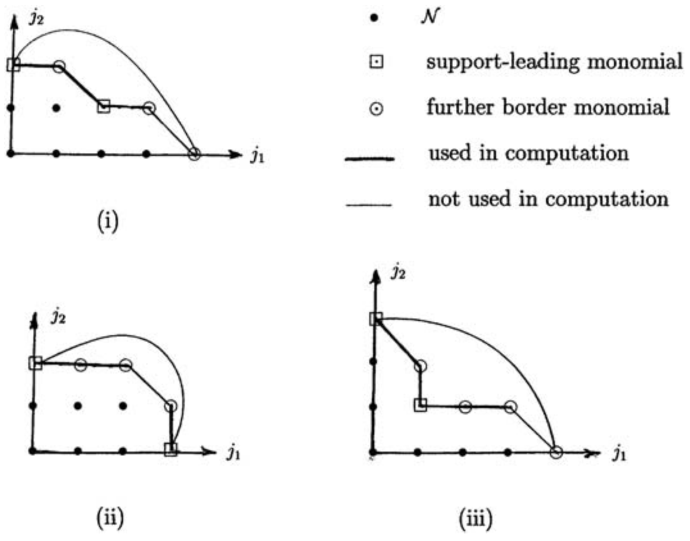


FIGURE 3

5. Algorithmic Determination of Border Bases

We assume that we have selected a tentative normal set $\mathcal{N}$, with its border web $BW_{\mathcal{N}}$, for a specified 0-dimensional polynomial system $P \subset \mathbb{P}^s$. Now we want to compute the coefficients of the associated border basis $\mathcal{B}_{\mathcal{N}} = \{bb_\nu,\, \nu = 1(1)n\}$ for the ideal generated by P.

By the selection procedure of Sect. 4, s of the bb_ν are specified by the polynomials of the autoreduced system P (we keep the notation p_ν for these polynomials): the support-leading monomials x^{ℓ_ν} of these p_ν are in the border set $B[\mathcal{N}]$. Thus it is natural to start the computation with syzygy relations (4) along edges of $BW_{\mathcal{N}}$ issuing from these monomials x^{ℓ_ν}, $\nu = 1(1)s$.

Assume at first that there is an edge of type (i) issuing from a particular one of these support-leading x^{ℓ_ν} in the σ'-direction and that the σ'-shifted truncated support set $x_{\sigma'}\bar{S}_\nu$ of $p_\nu = bb_\nu = x^{\ell_\nu} - a_{\ell_\nu}^T \mathbf{b}$ remains *completely within* $\mathcal{N} \cup \{x^{\ell_{\nu'}},\ \nu' = 1(1)s\}$. (This is most likely to happen for a p_ν of lowest degree.) Then, after the substitution of the occurring $x^{\ell_{\nu'}}$, $a_{\ell_\nu}^T x_{\sigma'}\mathbf{b}$ is a polynomial in $\mathcal{N}$ and must *equal* the polynomial $a_{k_{\nu'}}\mathbf{b}$ for $x^{k_{\nu'}} = x_{\sigma'}x^{\ell_\nu}$; cf. (4)(i). Thus, the border basis polynomial $bb_{k_{\nu'}}$ is obtained directly as $bb_{k_{\nu'}} := x^{k_{\nu'}} - a_{\ell_\nu}^T x_{\sigma'}\mathbf{b}$. Obviously, as a first step in our computation, we should attempt to utilize all possibilities of this kind.

After this initial phase, which may not be present at all, we must utilize the relations (4) along the other edges issuing from the x^{ℓ_ν}, $\nu = 1(1)s$. Each edge to a type (i) neighbor (cf. (3)) will generate a linear vector equation. If some pair of support-leading monomials is connected by a type (ii) edge, the associated relation (4)(ii) also leads to a linear vector equation. Generally, these equations will involve further vectors $a_{k_\nu}^T$ associated with border monomials x^{k_ν} introduced by the shifts of the x_{ℓ_ν}. Hopefully, there will be as many vector equations as unknown vectors. Note that edges issuing from border web nodes whose vectors have been found in the initial phase may be employed in the same fashion.

After more vectors $a_{k_\nu}^T$ have been found from this block of linear vector equations, the procedure may be further continued. But in choosing further edges we must watch that the emerging web of used edges becomes a *minimal* border web. This requests mainly that we do not use edges which close a loop; otherwise, we would introduce an equation which is dependent on equations which have been used already. With floating-point computation, this may not be realized and spoil the computation.

Hopefully, we may thus recursively generate *blocks* of linear vector equations, with as many equations as unknown vectors each, until all $N - s$ unknown vectors $a_{k_\nu}^T$ have been determined. In principle, this should always be possible, at least for a cleverly chosen normal set $\mathcal{N}$:

From the theory of Gröbner bases [2] we know that, for a regular 0-dimensional system P, the Gröbner basis w.r.t. any term order may be computed by a finite number of *rational* operations. It is clear that this holds also for the border basis associated with the normal set of a Gröbner basis. For a fixed system P, the change from the border basis polynomials for one normal set to those for another one also requires the solution of linear systems only; cf. [14, Sect. 8.1.1]. Therefore, the coefficients of *any* border basis for a regular 0-dimensional system P are *rational functions* of the coefficients of P. This raises some hope that border bases may generally be computed along the lines indicated, possibly with a more refined strategy. Note that — like in the F_5 algorithm of [3] — no *reductions to zero* will ever arise in our algorithm. These have been exposed as the main obstacles for a stable floating-point implementation of GB-algorithms; cf. [9].

Before we turn to the discussion of numerical difficulties which may arise even when we have recursive blocks of the correct number of linear equations, we continue our simple-minded example of Sect. 4 to explain the described algorithmic procedure in more detail. For a less trivial example, we refer to Example 7.2 and Fig. 5 (i).

Example 5.1. The original polynomials p_1, p_2 are dense in 2 variables, of degrees 2 and 3 resp., and $m = 6$; cf. Example 4.1.

(i) With the term order $\mathtt{tdeg}[x_2, x_1]$. we have obtained the normal set and border web of Fig. 3 (i), with x_2^2 and $x_1^2 x_2$ as final support-leading border monomials with specified coefficient vectors, and the further 3 border monomials $x_1 x_2^2$, $x_1^3 x_2$, and x_1^4 whose associated coefficient vectors have to be determined. For a clear notation, we set $\mathbf{b} := (1, x_1, x_2, x_1^2, x_1 x_2, x_1^3)^T$; then, with $a_k^T := (\alpha_{k,1}, \alpha_{k,2}, \alpha_{k,3},$
$\alpha_{k,4}, \alpha_{k,5}, \alpha_{k,6},)$, $a_k^T\,\mathbf{b}$ denotes the polynomial $\alpha_{k,1} + \alpha_{k,2}\,x_1 + \alpha_{k,3}\,x_2 + \alpha_{k,4}\,x_1^2$ $+ \alpha_{k,5}\,x_1 x_2 + \alpha_{k,6}\,x_1^3$.
In the initial phase of the algorithm, we see that the x_1-shift of the truncated support set $\bar{S}_1$ generates only normal set monomials and the border monomial $x_1^2 x_2$ whose coefficient vector $a_{(2,1)}^T$ is known from the autoreduced p_2. Hence, the substitution of $a_{(2,1)}^T$ into $a_{(0,2)}^T\,x_1 \mathbf{b}$ yields $a_{(1,2)}^T$ explicitly:

$$a_{(1,2)}^T := (0,\ \alpha_{(0,2),1}, 0,\ \alpha_{(0,2),2},\ \alpha_{(0,2),3},\ \alpha_{(0,2),4}) + \alpha_{(0,2),5}\, a_{(2,1)}^T\,.$$

For the simultaneous computation of $a_{(3,1)}^T$ and $a_{(4,0)}^T$, we may use the type (i) relation from an x_1-shift of $x_1^2 x_2$ together with the type (ii) relation between $x_1 x_2^2$ and $x_1^2 x_2$, both of which introduce the vectors $a_{(3,1)}^T$ and $a_{(4,0)}^T$ linearly, with known scalar coefficients:

$$\mathbf{a}_{(\mathbf{3},\mathbf{1})}^{\mathbf{T}} = (0,\ \alpha_{(2,1),1},\ 0,\ \alpha_{(2,1),2},\ \alpha_{(2,1),3},\ \alpha_{(2,1),4}) + \alpha_{(2,1),5},\ a_{(2,1)}^T + \alpha_{(2,1),6},\ \mathbf{a}_{(\mathbf{4},\mathbf{0})}{}^T\ ,$$
$$(0,\ \alpha_{(1,2),1}, 0,\ \alpha_{(1,2),2},\ \alpha_{(1,2),3},\ \alpha_{(1,2),4}) + \alpha_{(1,2),5}\, a_{(2,1)}^T + \alpha_{(1,2),6}\, \mathbf{a}_{(\mathbf{4},\mathbf{0})}{}^T =$$
$$(0, 0, \alpha_{(2,1),1}, 0, \alpha_{(2,1),2}, 0) + \alpha_{(2,1),3} a_{(0,2)}^T + \alpha_{(2,1),4} a_{(2,1)}^T + \alpha_{(2,1),5} a_{(1,2)}^T$$
$$+ \alpha_{(2,1),6} \mathbf{a}_{(\mathbf{3},\mathbf{1})}{}^T.$$

Thus, the complete computation of the Gröbner basis for $\mathtt{tdeg}[x_2, x_1]$ consists of substitutions and the solution of a 2×2 linear system with the matrix $\begin{pmatrix} 1 & -\alpha_{(2,1),6} \\ \alpha_{(2,1),6} & -\alpha_{(1,2),6} \end{pmatrix}$. It is hard to believe that the GB could be computed otherwise with fewer operations. (The reduced border web underlying this computation is shown in Fig. 3 (i). The case when the matrix is (near-) singular will be treated in the next section.)

(ii) With the rectangular normal set of Fig. 3 (ii), we have 3 type (i) edges and one type (ii) edge; thus, it is possible to use only type (i) relations for the computation of the 3 unknown vectors $a^T_{(1,2)}$, $a^T_{(2,2)}$ and $a^T_{(3,1)}$. The computation is very similar to the one for the preceeding normal set:
Again, the x_1-shift of the truncated support of p_1 introduces only normal set monomials and x_1^3 with its known vector $a^T_{(3,0)}$ into the expression for $a^T_{(1,2)}$. A simultaneous x_1-shift of $x_1x_2^2$ and x_2-shift of x_1^3 yields the two linear vector equations for $a^T_{(2,2)}$ and $a^T_{(3,1)}$. The manipulations are slightly simpler than in the previous case. The associated reduced border web (cf. Fig. 3 (ii)) remains connected by the virtual edge from x_2^2 to x_1^3.

(iii) With the unusual normal set of Fig. 3 (iii), the initial phase of the algorithm is empty. With simultaneous x_1- and x_2-shifts of the support-leading monomial x_1x_2, we obtain the system of linear equations for $a^T_{(2,1)}$ and $a^T_{(1,2)}$. Then we may use the type (ii) relation for $x_1x_2^2$ and x_2^3 together with the x_1-shift of $x_1^2x_2$ to obtain the two vector equations for $a^T_{(4,0)}$ and $a^T_{(3,1)}$. The lonely node x_1^4 is connected to the web by the virtual edge from x_1^4 to x_2^3; cf. Fig. 3 (iii).

6. Ill-Conditioned Situations

As explained in the Introduction, we assume throughout that the coefficients in the regular 0-dimensional polynomial system $P = \{p_\nu,\ \nu = 1(1)s\}$ are of *limited accuracy* and that the computation of a border basis for $\mathcal{I}[P]$ proceeds in *floating-point arithmetic*. Therefore, we must take special care to avoid *ill-conditioned* situations where small perturbations may be amplified excessively.

At first, the specified system P itself may be ill-conditioned, i.e. it may specify the ideal $\mathcal{I}[P]$ very poorly, with a high sensitivity to tiny changes in P. In a linear system, this happens when the (linear) polynomials are nearly linearly dependent or — equivalently — the system is close to singularity. The corresponding situation with polynomial systems will be discussed in the next section.

Now assume that P, with m joint zeros, is *not* an ill-conditioned system so that well-conditioned representations of $\mathcal{R}[\mathcal{I}[P]]$ must exist. Yet there may be closed convex sets of m monomials which represent $\mathcal{R}$ in an extremely sensitive fashion and thus are ill-suited as a basis for $\mathcal{R}$. How this may happen is easily seen:

For a specified normal set $\mathcal{N}$ and normal set vector $\mathbf{b}(x)$, we have (cf. (2))

$$bb_\nu \;=\; x^{k_\nu} - a^T_{k_\nu}\,\mathbf{b}(x) \;\in\; \mathcal{I}[P]\,, \quad \nu = 1(1)N\,,$$

which implies, for each ν,

$$z_\mu^{k_\nu} \;=\; a^T_{k_\nu}\,\mathbf{b}(z_\mu)\,, \quad \mu = 1(1)m\,, \tag{7}$$

where the z_μ are the m zeros of P. The extension to multiple zeros is straightforward but quite technical; we refer the reader to [14, Sect. 8.5]. By (7), the coefficient vectors $a_{k_\nu}^T$ can only be well-defined if the matrix

$$\mathbf{b}(z) := \begin{pmatrix} | & & & | \\ \mathbf{b}(z_1) & \dots & \dots & \mathbf{b}(z_m) \\ | & & & | \end{pmatrix} \in \mathbb{C}^{m\times m} \tag{8}$$

is well-conditioned. Clearly, the condition of $\mathbf{b}(z)$ depends strongly on the relation between the zero set $\{z_\mu,\ \mu = 1(1)m\}$ and the chosen normal set $\mathcal{N}$; it may differ considerably for different m-element normal sets.

Since the zeros z_μ are unknown at the time of the computation of the $a_{k_\nu}^T$, we cannot form $\mathbf{b}(z)$ explicitly. But we must take an ill-conditioning which appears in the computation of some $a_{k_\nu}^T$ as indication of an ill-conditioned $\mathbf{b}(z)$ and hence of an ill-suited normal set $\mathcal{N}$.

An ill-conditioning can already appear during the *autoreduction* phase through the choice of the support-leading monomials which strongly determine the normal sets which are admissible; cf. Sect. 4. Due to the dominant role which they play in the subsequent computation of the remaining $a_{k_\nu}^T$, the coefficient of the final support-leading monomial x^{ℓ_ν} of each autoreduced polynomial p_ν, $\nu = 1(1)s$, must not be very small relative to the coefficients in that p_ν.

If one or several of these coefficients should be tiny, we should change the (possibly recursive) selection of the support-leading monomials. Often it will be discernible which choice has introduced the ill-conditioning. In any case, the number of possible selections is generally quite limited.

Example 6.1. Assume the situation of Examples 4.1/5.1. We now specify the highest order terms of p_1 and p_2:

$$\begin{aligned} p_1(x_1,x_2) &= -.75\,x_1^2 + 2.1\,x_1x_2 + x_2^2 + \dots, \\ p_2(x_1,x_2) &= x_1^3 - .3\,x_1^2x_2 + 4.1\,x_1x_2 + 2.6\,x_2^3 + \dots; \end{aligned}$$

we aim for the final support-leading monomials of version (ii) of these examples which lead to the normal set of Fig. 3 (ii). However, reduction of p_2 by p_1 as in Example 4.1 (ii) leads to a p_2 with the 3rd order terms $-.02\,x_1^3 - 1.206\,x_1^2x_2$, where the coefficient of the proposed support-leading monomial x_1^3 is quite small. Here, the relief is straightforward: for the same final p_2, we use $x_1^2x_2$ as support-leading monomial which leads to the normal set of Fig. 3 (i).

During the computation proper, ill-conditioning may appear in the numerical solution of the blocks of linear equations for groups of $a_{k_\nu}^T$; cf. Example 5.1. This means that the relations (4) entering into that linear system are near-dependent. If they have arisen from edges of a *minimal* border web, this indicates that the selected normal set $\mathcal{N}$ furnishes an ill-conditioned basis of the quotient ring $\mathcal{R}[\mathcal{I}[P]]$ and that we should switch to another normal set. Naturally, we want to preserve as much as possible of the previous computation in that switch. This may generally be achieved in the following way:

We append one of the current border monomials to the normal set in exchange for one of the current normal set monomials. As candidates for this "degradation" we consider, at first, those border monomials which have figured in a relation (4) of the ill-conditioned block of equations. But the switch requires that the inclusion of that border monomial does not violate the *closedness* of $\mathcal{N}$; this means that it must not be a multiple of another border element. Usually, this decreases the number of candidates considerably; in some cases, the candidate set must even be extended to border monomials involved in an earlier step of the algorithm.

Similarly, the normal set monomial to be "upgraded to border" must not possess a multiple in the new normal set $\mathcal{N}'$. Of course, the switch may also generate further new border elements whose coefficient vectors, in the new basis $\mathcal{N}'$, have to be computed. But before we proceed to do this, we must *rewrite* the previously computed coefficient vectors into the new basis $\mathcal{N}'$, with normal set vector $\mathbf{b}'$.

Instead of an explanation of this rewriting procedure in general terms, we explain it in the context of our previous simple example. This will make it clear how to proceed in more realistic cases.

Example 6.2. Take the situation of Example 4.1/5.1 version (i) and consider the second step which yields the 2 equations for the coefficient vectors of $x_1^3x_2$ and x_1^4. Assume that this 2×2 system is ill-conditioned, i.e. that $\alpha_{(1,2),6} \approx \alpha^2_{(2,1),6}$. A scrutiny of the normal set of Fig. 3 (i) shows that only one of the 3 natural candidates $x_1x_2^2$, $x_1^2x_2$ and $x_1^3x_2$ is not a multiple of another border monomial, viz. $x_1^2x_2$. Also, the only normal set element without a multiple in the new normal set $\mathcal{N}'$ is x_1^3 which therefore becomes a border element. $\mathcal{N}'$ is now the normal set of Fig. 3 (ii) with the normal set vector $\mathbf{b}' = (1, x_1, x_2, x_1^2, x_1x_2, x_1^2x_2)^T$ which differs from $\mathbf{b} = (1, x_1, x_2, x_1^2, x_1x_2, x_1^3)^T$ only in the last (6th) component.

Therefore we introduce the truncated vector $\hat{\mathbf{b}} := (1, x_1, x_2, x_1^2, x_1x_2, 0)^T$ and write the polynomials $a_j^T\,\mathbf{b}$ as $aj^T\,\hat{\mathbf{b}} + \alpha_{j,6}x_1^3$. Thus, all we need is a representation of the new border monomial x_1^3 in terms of $\hat{\mathbf{b}}$ and the new normal set monomial $x_1^2x_2$. It is obtained by the inversion of

$$x_1^2x_2 \;=\; a_{(2,1)}^T\,\mathbf{b} \;=\; a_{(2,1)}^T\,\hat{\mathbf{b}} + \alpha_{(2,1),6}\,x_1^3$$

$$\text{into} \qquad x_1^3 \;=\; \frac{1}{\alpha_{(2,1),6}}\,[-a_{(2,1)}^T\,\hat{\mathbf{b}} + x_1^2x_2\,]\,.$$

Thus the original representation of the border element $x_1x_2^2$

$$x_1x_2^2 \;\equiv\; a_{(1,2)}^T\,\hat{\mathbf{b}} + \alpha_{(1,2),6}\,x_1^3 \qquad \text{becomes}$$

$$x_1x_2^2 \;\equiv\; (\,a_{(1,2)}^T - \frac{1}{\alpha_{(2,1),6}}\,a_{(2,1)}^T\,)\,\hat{\mathbf{b}} + \frac{\alpha_{(1,2),6}}{\alpha_{(2,1),6}}\,x_1^2x_2 \;=:\; (a'_{(1,2)})^T\,\mathbf{b}'\,.$$

Of course, in a more realistic situation, further adaptations of the normal set may be necessary to reach a well-conditioned basis for $\mathcal{R}[\mathcal{I}[P]]$ which must exist if P is well-conditioned. Also, if *no* ill-conditioned blocks arise during a border basis computation, we may conclude that the computed border basis $\mathcal{B}_{\mathcal{N}}$ is a well-conditioned basis for $\mathcal{I}[P]$.

7. Near-Singular Situations

A polynomial system with as many polynomials as variables is called *singular* if it

- either has (only isolated) but *fewer zeros* than its BKK-number m indicates,
- or it possesses a *zero manifold*.

This agrees with the usage for *linear* systems where the BKK-number is always 1; there it is also obvious that both cases are only the two sides of the same coin (singular matrix). With exact polynomials, either situation can only arise for very special precise values of the coefficients. Numerically, like in numerical linear algebra, one will generally meet only near-singular systems which are *very close* to an exact singular system.

Treated as exact systems, such near-singular systems are extremely ill-conditioned. The rapid movements of some of the m disjoint zeros during an assumed transition into an exactly singular system have been analyzed in [14, Sects. 9.4 and 9.5]. With *empirical systems* whose coefficients have a limited accuracy, it is generally more meaningful to assume that a very-nearly-singular systems stands for a nearby strictly singular one. With this taken into account, the ill-conditioning disappears and an algorithmic treatment becomes feasible. This is in analogy with the situation for dense clusters and multiple zeros: if a dense cluster of $\bar{m}$ zeros is treated as an $\bar{m}$-fold zero of a nearby system, the determination of this zero and of its structure becomes well-conditioned, cf. [5] and [14, Sect. 9.3].

Let us first consider the case of fewer than m zeros where some zeros have "diverged" to infinity, as was already remarked by D. Bernstein [1]. Here, for *any* m-element normal set, the attempt to compute a border basis must meet with a *singular, inconsistent* block of linear vector equations. If this fact is computationally well established, we may assume that we have a BKK-deficient system and treat it as such:

We consider a particular m-element normal set and the associated singular block which has arisen; we assume at first that its numerical rank deficiency is 1. Then there exists one linear combination of the linear vector equations which annihilates the terms with the unknown vectors and thus furnishes a *linear relation between the monomials of* $\mathcal{N}$. We select an element x^{j_μ} of $\mathcal{N}$ whose upgrading into border is feasible and detach it from $\mathcal{N}$ to generate an $(m-1)$-member normal set $\mathcal{N}'$; then we obtain the coefficient vector $(a')^T_{j_\mu}$ of x^{j_μ} w.r.t. $\mathcal{N}'$ by solving the linear relation in $\mathcal{N}$ for x^{j_μ}. Previously specified or computed coefficient vectors of border elements w.r.t. $\mathcal{N}$ are converted by the substitution of $(a')^T_{j_\mu}\,\mathbf{b}'$ for x^{j_μ} in their normal forms.

The computation may then be continued and — hopefully — completed, resulting in a border basis $\mathcal{B}_{\mathcal{N}'}$ w.r.t. $\mathcal{N}'$, whose $(m-1)\times(m-1)$ multiplication matrices furnish the $m-1$ zeros of a strictly BKK-deficient system P' very close to our specified system P. These zeros will generally be excellent approximations of the $m-1$ exact zeros of P with moderate modulus while the remaining exact zero lies extremely far from the origin and may therefore be considered as *diverged to* ∞ for practical purposes; cf. [14, Sect. 9.5].

A higher BKK-deficiency will become apparent either through a higher rank deficiency of some singular block or through a sequence of rank deficiencies 1 as above.

As previously, we explain details of the algorithmic procedure by demonstration with a simple example:

Example 7.1. Again we take $s = 2$ and polynomials of degrees 2 and 3 resp., but here we have to specify the coefficients: $P = \{p_1, p_2\}$, with

$$\begin{aligned} p_1(x_1, x_2) &:= 100\,x_1^2 - 15\,x_1x_2 - 76\,x_2^2 + 25\,x_1 - 15\,x_2 + 1\,, \\ p_2(x_1, x_2) &:= x_1^2x_2 - .95\,x_1x_2^2 - 2\,x_1^2 + x_1x_2 - .5\,x_2^2 + 4\,x_1 - 5\,x_2 + 3\,. \end{aligned}$$

The BKK-number of P is 6. The following computations have been performed with Maple 9.5, with `Digits:=10`.

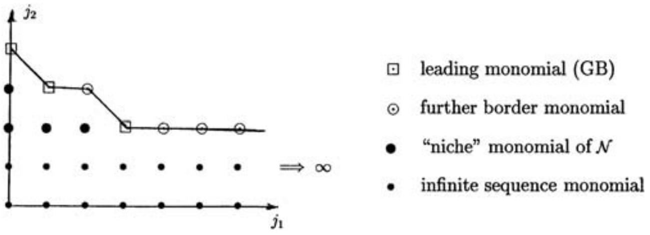
We autoreduce p_2 and choose $\mathcal{N}$ as in Example 4.1 (ii); then we proceed as in Example 5.1 (ii). But in the joint computation of $a_{(3,1)}^T$ and $a_{(2,2)}^T$ we meet a numerically singular matrix, with a determinant of $\mathrm{O}(10^{-10})$. The attempt to overcome the extreme ill-conditioning by a switch of monomials between $\mathcal{N}$ and $B[\mathcal{N}]$ as described in Sect. 6 brings no relief; the singularity perseveres. Thus we (correctly) assume that P is BKK-deficient, with deficiency 1.

The only detachable monomial of $\mathcal{N}$ is $x_1^2x_2$ for which we obtain a representation w.r.t $\mathcal{N}' := \mathcal{N} \setminus \{x_1^2x_2\}$ from the linear relation between the right-hand sides of the singular system. Thus, instead of determining the vectors $a_{(3,1)}^T$ and $a_{(2,2)}^T$, the singular 2-block has determined the *one* vector $(a')_{(2,1)}^T$ only. It remains to substitute $(a')_{(2,1)}^T\,\mathbf{b}'$ into the representation of the other $\mathcal{N}'$-border monomials. This completes the task.

The 5 zeros computed from the eigenvectors of the associated multiplication matrices are very good approximations of the exact zeros of the autoreduced system P. With the use of high accuracy, the missing 6th zero of this very-near-singular system is discovered at approx. $(1.1\cdot10^{10}, 1.1\cdot10^{10})$ which may be regarded as ∞.

Let us now consider the case where an apparently regular system, with BKK-number m, possesses a *zero manifold* of dimension $d > 0$ while systems arbitrarily close to it have only the m isolated zeros (counting multiplicities). The analysis of what happens when the coefficients of such a system move from regular to singular values is very interesting (cf. [14, Sect. 9.4]); but we will not discuss it here. In the context of this paper, we want to understand how this type of singularity manifests itself in the computation of a border basis so that we may recognize a very-near singular system of this kind. In applications, such systems which are extremely ill-conditioned should be identified with a nearby strictly singular system. Often, the fact that a situation admits a manifold of solutions may be the most important result of the analysis of the model.

For a positive-dimensional polynomial system P, the associated quotient ring has vector dimension *infinity* so that a monomial basis must have infinitely many elements. In our visualization of normal sets by the exponents of the monomials, it means that these exponents most cover a complete subspace of dimension d. Fig. 4 shows a typical normal set for a 1-dimensional system with 2 variables; the 4 monomials *above* the infinite sequences reveal the existence of 4 isolated zeros besides the manifold.



j_2
j_1
$\Rightarrow \infty$
⊡ leading monomial (GB)
⊙ further border monomial
● "niche" monomial of $\mathcal{N}$
• infinite sequence monomial

FIGURE 4

The Gröbner basis for this normal set consists of 3 polynomials, with leading monomials $x_2^4, x_1x_2^2, x_1^2x_2$; the absence of a leading monomial on the x_1-axis indicates the 1-dimensionality. Multiplication matrices for this basis formed in the usual manner would have infinitely many rows and columns; but it can be shown that a finite rectangular section of the matrices contains the complete information about the zeros; cf. [14, Chapter 11]. It is not yet clear how border bases could be defined in a meaningful way in this situation; cf. conclusions in [6].

In such a system, we should hit upon a block of linear vector equations which is *singular but consistent.* This means that the coefficient vectors $a_{k_\nu}^T$ of one of the border monomials x^{k_ν} figuring in that block may be moved into the normal set *without* simultaneous conversion of a normal set monomial into a border monomial. For a 0-dimensional system, there cannot exist *more* than m normal set monomials; hence the system must actually be positive-dimensional!

With the availability of further elements in the normal set, it should now be possible to extend $\mathcal{N}$ further and further in a coordinate direction. Very soon, this computation will become recursive and need not be continued. The coefficient vectors for other remaining border monomials should be computable in the normal fashion. A complete analysis of this situation is still in the future. We will again restrict ourselves to the discussion of an example.

Example 7.2. We consider the system

$$\begin{aligned}
p_1(x_1,x_2,x_3) &:= x_1^2 + x_1x_2 - x_1x_3 - x_1 - x_2 + x_3\,, \\
p_2(x_1,x_2,x_3) &:= x_1x_2 + c\,x_2^2 - x_2x_3 - x_1 - c\,x_2 + x_3\,, \\
p_3(x_1,x_2,x_3) &:= x_1x_3 + x_2x_3 - x_3^2 - x_1 - x_2 + x_3\,;
\end{aligned}$$

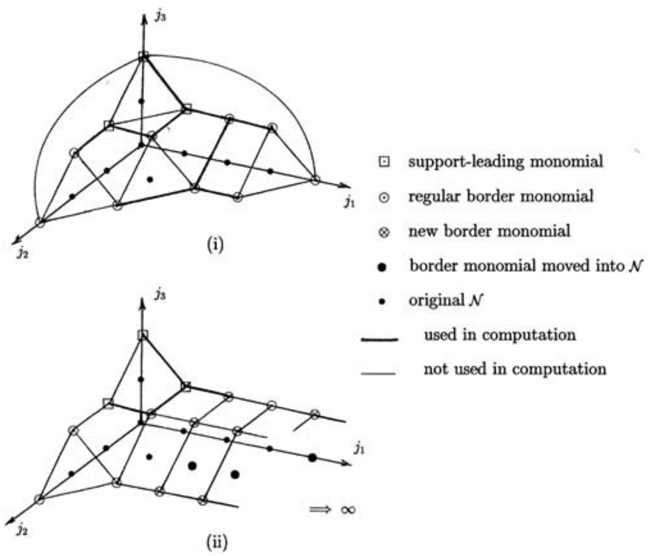


FIGURE 5

the parameter c remains indeterminate at first. We select the support-leading monomials according to the $\mathtt{tdeg}(x_3, x_2, x_1)$ order as $x_1x_3,\ x_2x_3,\ x_3^2$ resp. With $m{=}8$, we obtain the (apparent) normal set $\mathcal{N} := \{1,\ x_1,\ x_2,\ x_3,\ x_1^2,\ x_1x_2,\ x_2^2,\ x_1^3\}$, with a border set of 12 monomials. The complete border web has 21 edges, only $12-3=9$ of which should figure in a minimal web; cf. Fig. 5 (i). Until now, there is now immediate sign of the singularity.

Starting with the support-leading nodes, we obtain the following 4 by 4-block for $a^T_{(2,1,0)}, a^T_{(1,2,0)}, a^T_{(2,0,1)}, a^T_{(1,1,1)}$ from the type (ii) relation (4) for the edge $[x_3^2, x_1x_3]$ and the type (i) relations along the edges $[x_1x_3, x_1^2x_3]$, $[x_1x_3, x_1x_2x_3]$ and $[x_2x_3, x_1x_2x_3]$:

$$\begin{pmatrix} -2 & -c & 1 & 1 \\ -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \\ -1 & -c & 0 & 1 \end{pmatrix} \begin{pmatrix} a^T_{(2,1,0)} \\ a^T_{(1,2,0)} \\ a^T_{(2,0,1)} \\ a^T_{(1,1,1)} \end{pmatrix} \mathbf{b}$$

$$= \begin{pmatrix} -2 & -2 & 2 & 0 & 1-c & 0 & 1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 1 \\ -1 & -c & 1 & 0 & 0 & c-1 & 0 \\ -1 & -1 & 1 & 0 & 1-c & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \\ x_1^3 \end{pmatrix}.$$

It is easily seen that both sides of the equation are annihilated by pre-multiplication with $(-1,\ 1,\ 0,\ 1)$, for any value of the parameter c; therefore, we actually have a

1-dimensional singular system. (In this simple example, it is also not difficult to see that there are two lines of zeros: $x_1 = x_3$, $x_2 = 0$ and $x_1 = 1 - x_3$, $x_2 = 1$ for all values of c, with special situations for $c = 0$ and $c = 1$.)

Of the border monomials in the above equation, only $x_1^2x_2$ and $x_1x_2^2$ are eligible for transfer into the normal set. We choose $x_1^2x_2$ and can now determine the coefficient vectors $a_{(1,2,0)}^T$, $a_{(2,0,1)}^T$ and $a_{(1,1,1)}^T$ uniquely (except for $c = 1$) in terms of the enlarged normal set. Also, we have two new border monomials: $x_1^2x_2^2$ and $x_1^2x_2x_3$. It turns out that $a_{(2,1,1)}^T$ can only be computed if the potential border monomial $x_1^3x_2$ is also included into the normal set, cf. Fig. 5 (ii). Thus, the situation becomes recursive and will continue to infinity.

Note that a border basis for the same system, with generically perturbed coefficients, can readily be computed for the normal set chosen above; cf. Fig. 5 (i).

Numerically, the *very-near-singular* case will be more important. It is now clear that it may be recognized by a block which is a tiny perturbation of a singular consistent set of vector equations. An exact solution would be extremely sensitive and therefore ill-determined in this case; actually, except in the case of confluence, some of the exact isolated zeros of a very-near-singular system of this kind may approach *any* point on the singular manifold in the transition to strict singularity (cf. [14, Sect. 4])! Therefore, the determination of that manifold is generally far more relevant than the computation of “exact” zeros.

8. Implementation

It is a main purpose of this publication to stimulate attempts towards an implementation of the algorithmic procedure described in the previous sections. This will permit large-scale experimentation on non-trivial problems which, in turn, will lead to a preliminary assessment of the potential efficiency of our approach and of its numerical stability.

An implementation will have to proceed in two stages:

In the first stage, exact data and exact (rational) computation will be assumed and used. Thus, the emphasis will lie on the automatization of the decisions which have to be made to determine the algorithmic flow; numerical stability will not be a topic in this stage. In the second stage, floating-point data and floating-point computation will be assumed so that the numerical condition of the individual steps has to be checked; this will introduce further restrictions on the choices and also potential changes in the algorithmic flow.

It is the overall strategic goal of the organizational phase of our algorithm to generate a normal set whose border web permits the recursive selection of sets of syzygies which lead to blocks of linear vector equations for as many coefficient vectors $a_{j_\nu}^T$ as there are equations, until the representations of all border monomials have been found. From Sect. 2, we remember that *linear* equations are generated

- either by type (i) edges $[x^{j_\nu}, x^{j_\nu+e_\sigma}]$, with $a^T_{j_\nu}$ known[1],

- or by type (ii) edges $[x^{k_\mu+e_{\sigma_1}}, x^{k_\mu+e_{\sigma_2}}]$, $x^{k_\mu} \in \mathcal{N}$, with $a^T_{k_\mu+e_{\sigma_1}}$ *and* $a^T_{k_\mu+e_{\sigma_2}}$ known.

Thus, our computation can only get started, if there are type (i) edges issuing from the support-leading nodes and/or type (ii) edges connecting two support-leading nodes. Also, the overall occurrence of many type (i) edges in the web should help in the continuation of the computation. It will require some ingenuity to build a strategy for an optimal satisfaction of these requirements into the initial phase of the algorithm which chooses the support-leading monomials for the autoreduction, and then selects an m-element closed normal set consistent with the supports and the final support-leading monomials.

Besides, it is an open question whether there do exist normal sets for *arbitrary* regular polynomial systems which support the computation of the complete border basis by our approach; efforts towards an implementation may help to answer that question. Since we *know* from GB theory that there exist normal sets with border bases whose coefficients are rational functions of the data, a positive answer does not appear impossible.

Naturally, the unfamiliar choices which have to be made stem from the fact that we work *without a term order*. Actually, this is one of the major attractions of our approach, particularly when numerical stability is also an issue. As remarked in Sect. 4, when we choose the support-leading monomials as leading monomials w.r.t. a term order, we will generally obtain the reduced GB for that term order as the "corner subset" of the border basis, perhaps after enforced changes in the initial normal set. With this restricted set of normal sets, an answer to the question whether our approach will always succeed, or for which classes of systems, would be of particular interest. In Example 7.2, we have employed the term order $\mathtt{tdeg}(x_3, x_2, x_1)$; our algorithm has worked in the singular as well as in the regular case (with perturbed coefficients) and generated the GB.

While the selection of syzygies for the beginning of the computation is determined by the support-leading monomials with their specified normal forms, the further continuation of the computation is not always clear. Probably, for an algorithm, one should simply form all linear syzygies whose edges do not close loops, and then look for blocks with as many equations as unknown vectors. If there is no such block, the algorithm cannot be continued. In this case, a feasible exchange between a normal set and a border monomial may often open the deadlock so that a full restart with a different normal set is not necessary. How to check that algorithmically, with a meaningful strategy, is not clear.

In the second stage of the design of an implementation of our approach, an algorithm which complies with all the considerations so far must now be adapted to the use of floating-point arithmetic. To my knowledge, none of the presently available GB-packages can handle that situation (or only with a ridiculous digit

[1]Here, e_σ is the σ-th unit vector.

swell, when floating-point data are converted to rational data and exact computation is then employed). But models of real-life situations have floating-point data, with a limited accuracy, in almost all cases!

Now, the considerations in Sects. 6 and 7 must also be implemented. As usual, the threshold for what is called ill-conditioned is arbitrary to some extent: with a 10 digit computation, e.g., relative condition numbers should supposedly not exceed 10^6 or so. If a computation shows signs of numerical instability, the floating-point accuracy should be raised with care (the algorithm must provide for that). When the results do not agree for higher accuracies, the system itself probably defines its ideal in such an ill-conditioned fashion that a determination is not meaningful.

The algorithmic checking of the relative sizes of data must begin with the selection of the support-leading terms; their coefficients must not be much smaller in modulus than those of the remaining terms. If there is only one highest degree term with a very small coefficient, one must attempt to get rid of it during autoreduction; if this is not possible the specified system itself is ill-conditioned.

During the further computation, the condition of the linear equation blocks must be checked, and the exchange mechanism of Sect. 6 must be used if a condition number is too high relative to the floating-point accuracy. Now there is generally a very small set of candidates for both parts in the exchange; if there is still a choice, the generation of fewer or more computable edges may be used as a criterion.

This holds also for the cases where the ill-conditioning is not due to an ill-chosen normal set but to a near-singularity in the specified system. Now, the ill-conditioning will not disappear with an exchange, or the situation does not permit an exchange. One should then proceed as discussed in Sect. 7.

Once more it is an open question whether there may exist situations where the stabilized algorithm runs into a dead end while the instable computation would have succeeded. Note also that the exchange mechanism in Sect. 6 generally proceeds just like the “extension” mechanism which had been proposed by myself many years ago (cf. [13]); but now we are not violating the term order as with the “extended GB” approach because there is no term order which restricts us. This shows once more that numerical stability in the computation of an ideal basis can only be realized without a pre-specified term order.

9. Conclusions

I have tried to show how the use of the BKK-number opens the way for a new algorithmic approach to the numerical computation of a border basis for a regular polynomial system. This approach employs *no term order* and *no reductions to zero*. The first feature permits more flexibility in the representation of the quotient ring while the second feature permits a stable implementation in floating-point arithmetic. Both features should make the approach attractive.

It should be interesting to compare this approach with the one of [12] and to analyze the similarities and differences. In the approach of [12], one begins with a tentative normal set which is generally too large and has to be successively reduced during the computation of normal forms for border elements.

No serious implementation of my approach exists so far; it is hoped that this publication may lead to implementations in the near future. Only then, the potential of the approach may seriously be assessed. In particular, it should then become clear whether the approach can be made to work for arbitrary regular systems or only for certain subclasses of such systems and how its efficiency compares with other recent approaches.

In any case, on the way to an implementation new insights about border bases should come to light, with potential applications in other parts of numerical polynomial algebra.

References

[1] D. Bernstein: The Number of Roots of a System of Equations, Funct. Anal. Appl. **9**(3) (1975) 183–185.

[2] B. Buchberger: A Criterion for Detecting Unnecessary Reductions in Polynomial Ideal Theory, in: Recent Trends in Multidimensional System Theory (Ed. N. K. Bose), Chapter 6, 184–232. D. Reidel Publ. Co., 1986.

[3] J.-C. Faugère: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F_5), in: Proc. ISSAC 2002 (Ed. C. Traverso), 75–83, ACM Press, New York, 2002.

[4] B. Huber, B. Sturmfels: Bernstein's Theorem in Affine Space, Discr. Comput. Geom. **17** (1997) 137–141.

[5] W. Kahan: Conserving Confluence Curbs Ill-Condition, Dept. Comp. Sci., Univ. Calif. Berkley, Tech. Rep. 6, 1972.

[6] A. Kehrein, M. Kreuzer: Characterization of Border Bases, J. Pure Appl. Algebra **196** (2005) 251–270.

[7] A. Kehrein, M. Kreuzer: Computing Border Bases, J. Pure Appl. Algebra **205** (2006) 279–295.

[8] A. Kehrein, M. Kreuzer, L. Robbiano: An Algebraist View on Border Bases, in: Solving Polynomial Equations. Foundations, Algorithms, and Applications (Eds. A. Dickenstein, I. Emiris), 169–202, Springer, 2005.

[9] A. Kondratyev: Numerical Computation of Gröbner Bases, Ph.D. Thesis, Univ. of Linz, 359 pp., 2003.

[10] T. Y. Li: Numerical Solution of Polynomial Systems by Homotopy Continuation Methods, in: Handbook of Numerical Analysis, vol. XI (Ed. F. Cucker), 209–304, North-Holland, 2003.

[11] B. Mourrain: A New Criterion for Normal Form Algorithms, in: Lect. Notes Sci. Comp. **179**, 430–443, Springer, Berlin, 1999.

[12] B. Mourrain, P. Trébuchet: Generalized Normal Forms and Polynomial Systems Solving, in: Proc. ISSAC 2005 (Ed. M. Kauers), 253–260, ACM Press, New York, 2005.

[13] H. J. Stetter: Stabilization of Polynomial Systems Solving with Groebner Bases, in: Proc. ISSAC 1997 (Ed. W. Kuechlin), 117–124, ACM Press, New York, 1997.

[14] H. J. Stetter: *Numerical Polynomial Algebra*, XVI + 472 pp., SIAM Publ., Philadelphia, 2004.

[15] B. Sturmfels: Gröbner Bases and Convex Polytopes, AMS University Lecture Series vol 8, 1996.

[16] J. Verschelde: Algorithm 795: PHC-pack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation, Trans. Math. Software **25** (1999) 251–276.

Hans J. Stetter
Am Modenapark 13/4
A-1030 Vienna Austria
e-mail: `stetter@aurora.anum.tuwien.ac.at`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 269–278
© 2007 Birkhäuser Verlag Basel/Switzerland

Evaluation of Jacobian Matrices for Newton's Method with Deflation to Approximate Isolated Singular Solutions of Polynomial Systems

Anton Leykin, Jan Verschelde and Ailing Zhao

Abstract. For isolated singular solutions of polynomial systems, we can restore the quadratic convergence of Newton's method by deflation. The number of deflation stages is bounded by the multiplicity of the root. A preliminary implementation performs well in case only a few deflation stages are needed, but suffers from expression swell as the number of deflation stages grows. In this paper we describe how a directed acyclic graph of derivative operators guides an efficient evaluation of the Jacobian matrices produced by our deflation algorithm. We illustrate how the symbolic-numeric deflation algorithm can be used within *PHCmaple* interfacing Maple with PHCpack.

Mathematics Subject Classification (2000). Primary 65H10; Secondary 14Q99, 68W30.

Keywords. Deflation, evaluation, isolated singular solution, Jacobian matrix, Newton's method, numerical homotopy algorithm, polynomial system, reconditioning, symbolic-numeric computation.

1. Introduction

Newton's method slows down when approaching a singular root and convergence may even be lost if the working precision is not high enough. Moreover, using multiprecision arithmetic makes sense only if the input data is sufficiently precise. To restore the quadratic convergence of Newton's method, T. Ojika, S. Watanabe, and T. Mitsui ([11], see also [10]) developed a numerical deflation algorithm. A symbolic algorithm to restore the quadratic convergence of Newton's method was developed by Lecerf [6].

This material is based upon work supported by the National Science Foundation under Grant No. 0134611 and Grant No. 0410036.

We studied the methods proposed in [11] and [10] and – for the sake of numerical stability – implemented and experimented with two modifications to their deflation algorithm:

1. Instead of *replacing* equations of the original system by conditions from derivatives of the system, we propose to *add* equations, introducing random constants for uniform treatment.
2. Instead of using Gaussian Elimination, we propose to apply Singular Value Decomposition (SVD) to determine the numerical rank[1] of the Jacobian matrix. The threshold on the numerical rank is our only critical numerical parameter to decide whether to deflate or not to deflate.

In particular, if the numerical rank of the Jacobian matrix at the current approximation equals R, we introduce $R+1$ additional variables which serve as multipliers to selections of random combinations of columns of the Jacobian matrix (first presented in [18]). In [8] we showed that no more than $m-1$ successive deflations are needed to restore the quadratic convergence of Newton's method converging to an isolated root of multiplicity m. Once the precise location of the isolated singularity is known, numerical techniques allow the calculation of the multiplicity structure, using the methods in [1], [2], or [12] (see also [13, 16]).

The paper [2] analyzes our deflation algorithm from a different perspective, showing that our algorithm produces differentials in the dual space as a by-product. For the important special case when the Jacobian matrix has corank one, a modification of the deflation algorithm is presented in [2], which mitigates the expential growth of the size of the matrices.

Our modifications to the methods of [11] and [10] were first [18] developed in Maple, exploiting its facilities for polynomial manipulations and convenient multi-precision arithmetic, and then implemented in PHCpack [17]. While the performance of the method is promising on selected applications (such as the fourfold isolated roots of the cyclic 9-roots problem, see [8]), the method suffers from expression swell after a couple of deflations. In this paper, we describe a way to "unfold" the extra multiplier variables, exploiting the special structure of the matrices which arise in the deflation process. In the unfolding process, we naturally arrive at trees (or more precisely, directed acyclic graphs), also employed in [4, 5].

Our Jacobian matrices have a particular sparse block structure which should be also be exploited when computing the numerical rank and solving the linear system. For this, we recommend the recent rank-revealing algorithms in [9].

Our algorithm is a symbolic-numeric algorithm; perhaps, the term "numeric-symbolic" is more appropriate, as we produce by numerical means new equations which regularize or recondition the problem. In particular, our algorithm gives – in addition to more accurate values for the coordinates of the solution – a new system of polynomial equations for which the isolated singular solution is a regular root. Like [14], this paper documents the recent improvements to PHCpack [17].

[1]The determination of the numerical rank is a well studied problem in numerical linear algebra, for recent progress we refer to [3] and [9].

We encounter singular solutions when we solve polynomial systems using homotopy continuation methods, see e.g. [15]. This encounter can only happen – with probability one – at the end of the solution paths defined by the homotopy. So homotopy continuation methods deliver approximate solutions to the singularities which are then reconditioned by the deflation algorithm. One future project is the use of deflation to decide *locally* whether a solution at the end of a path is isolated or lies on a positive dimensional solution set. For this problem, deflation is needed because the solution set might be multiple.

2. Problem Statement

In this section we fix the notation. Consider $f(\mathbf{x}) = \mathbf{0}$, a system of N polynomial equations in n unknowns, with isolated multiple root $\mathbf{x}^*$. As we restrict ourselves to isolated roots, we have $N \geq n$. At $\mathbf{x} = \mathbf{x}^*$, the Jacobian matrix $A(\mathbf{x})$ of f has rank $R < n$. At stage k in the deflation, we have the system

$$f_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left\{ \begin{array}{rcl} f_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) & = & 0 \\ A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) B_k \boldsymbol{\lambda}_k & = & 0 \\ \mathbf{h}_k \boldsymbol{\lambda}_k & = & 1 \end{array} \right. \tag{1}$$

with $f_0 = f$, $A_0 = A$, and where $\boldsymbol{\lambda}_k$ is a vector of $R_k + 1$ multiplier variables, $R_k = \text{rank}(A_{k-1}(\mathbf{x}^*))$, scaled using a random vector $\mathbf{h}_k \in \mathbb{C}^{R_k+1}$. The matrix B_k is a random matrix with as many rows as the number of variables in the system f_{k-1} and with $R_k + 1$ columns. The Jacobian matrix of f_k is A_k. We have the following relations

$$\#\text{rows in } A_k: \; N_k = 2N_{k-1} + 1, \; N_0 = N, \tag{2}$$

$$\#\text{columns in } A_k: \; n_k = n_{k-1} + R_k + 1, \; n_0 = n. \tag{3}$$

The second line in (1) requires the multiplication of N_{k-1}-by-n_{k-1} polynomial matrix A_{k-1} with the random n_{k-1}-by-$R_k + 1$ matrix B_k, with the vector of $R_k + 1$ multiplier variables $\boldsymbol{\lambda}_k$.

If the evaluation of $A_{k-1} B_k \boldsymbol{\lambda}_k$ is done symbolically, i.e.: if we first compute the polynomial matrix $A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}) B_k$ before we give values to $(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1})$, the expression swell will cause the evaluation to be very expensive. In this paper we describe how to first evaluate the Jacobian matrices before the matrix multiplications are done. As this technical description forms a blueprint for an efficient implementation, it also sheds light on the complexity of the deflation.

3. Unwinding the Multipliers

We observe in (1) that the multiplier variables in $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_k$ all occur linearly. The Jacobian matrix of f_k in (1) has a nice block structure which already separates the linearly occurring $\boldsymbol{\lambda}_k$ from the other variables:

$$A_k(\mathbf{x}, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \left[\begin{array}{cc} A_{k-1} & \mathbf{0} \\ \left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} \cdots \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} \right] B_k \boldsymbol{\lambda}_k & A_{k-1} B_k \\ \mathbf{0} & \mathbf{h}_k \end{array} \right], \quad (4)$$

where the partial derivatives of an N-by-n matrix A with respect to a vector of variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ give rise to a vector of matrices:

$$\frac{\partial A}{\partial \mathbf{x}} = \left[\frac{\partial A}{\partial x_1} \; \frac{\partial A}{\partial x_2} \; \cdots \; \frac{\partial A}{\partial x_n} \right], \quad \frac{\partial A}{\partial x_k} = \left[\frac{\partial a_{ij}}{\partial x_k} \right] \text{ for } A = [a_{ij}], \quad \begin{array}{l} {\scriptstyle i=1,2,\ldots,N,} \\ {\scriptstyle j=1,2,\ldots,n,} \\ {\scriptstyle k=1,2,\ldots,n.} \end{array} \quad (5)$$

The definition of $\frac{\partial A}{\partial \mathbf{x}}$ naturally satisfies the relation $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$.

Notice that in (4) the evaluation of

$$\left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \; \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} \; \cdots \; \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} \right] B_k \boldsymbol{\lambda}_k \quad (6)$$

yields the matrix

$$\left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} B_k \boldsymbol{\lambda}_k \; \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} B_k \boldsymbol{\lambda}_k \; \cdots \; \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} B_k \boldsymbol{\lambda}_k \right] \quad (7)$$

which has the same number of columns as A_{k-1}.

As we started this section observing the separation of $\boldsymbol{\lambda}_k$ in the block structure of A_k, we can carry this further through to all multiplier variables. By "unwinding" the multipliers, we mean the removal of the $\frac{\partial}{\partial \boldsymbol{\lambda}}$-type derivatives. In the end, we will only be left with derivatives of the variables $\mathbf{x}$ which are the only variables which may occur nonlinearly in the given system f.

In the k-th deflation stage, we will then need $\frac{\partial A}{\partial \mathbf{x}}, \frac{\partial^2 A}{\partial \mathbf{x}^2}, \ldots, \frac{\partial^k A}{\partial \mathbf{x}^k}$. If we execute (5) blindly, with disregard of the symmetry, we end up with n^k matrices, e.g.: for $n = 3$ and $k = 10$, we compute 59,049 3-by-3 polynomial matrices, which is quite discouraging. However, as $\frac{\partial^2 A}{\partial x_1 \partial x_2} = \frac{\partial^2 A}{\partial x_2 \partial x_1}$, we enumerate all different monomials in n variables of degree k which is considerably less than n^k, e.g.: for $n = 3$ and $k = 10$ again, we now only have 66 distinct polynomial matrices to compute. To store $\frac{\partial A}{\partial \mathbf{x}}$, a tree with n children (some branches may be empty) is a natural data structure, see [4, 5].

4. A Directed Acyclic Graph of Jacobian Matrices

In this section, we explain by means of examples, how we build our data structures.

For example, consider $k = 2$:

$$A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = \left[\begin{array}{cc} A_1 & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \right] B_2 \boldsymbol{\lambda}_2 & A_1 B_2 \\ \mathbf{0} & \mathbf{h}_2 \end{array} \right]. \quad (8)$$

The evaluation of A_2 requires

$$A_1(\mathbf{x}, \boldsymbol{\lambda}_1) = \left[\begin{array}{cc} A & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 & A B_1 \\ \mathbf{0} & \mathbf{h}_1 \end{array} \right] \tag{9}$$

and the derivatives

$$\frac{\partial A_1}{\partial \mathbf{x}} = \left[\begin{array}{cc} \frac{\partial A}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2}\right] B_1 \boldsymbol{\lambda}_1 & \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \\ \mathbf{0} & \mathbf{0} \end{array} \right] \quad \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} = \left[\begin{array}{cc} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right]. \tag{10}$$

For $\frac{\partial A}{\partial \mathbf{x}}$ as in (5), the product $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1$ is $\left[\frac{\partial A}{\partial x_1} B_1 \ \frac{\partial A}{\partial x_2} B_1 \ \cdots \frac{\partial A}{\partial x_n} B_1\right]$. On the other hand, $\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1$ uses the operator $*$ which treats its second argument as a vector of columns, i.e. if $B_1 = (b_1, b_2, ..., b_m)$ where b_i $(1 \le i \le m = R_1 + 1)$ are the columns, then

$$\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 = \left[\frac{\partial A}{\partial \mathbf{x}} b_1 \ \frac{\partial A}{\partial \mathbf{x}} b_2 \ \cdots \frac{\partial A}{\partial \mathbf{x}} b_m\right]. \tag{11}$$

One may evaluate $\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1$ by computing the product $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1$ followed by alternatingly permuting the columns of the result. By virtue of this operator $*$, we may write

$$\frac{\partial}{\partial \boldsymbol{\lambda}_1} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 \right) = \left[\frac{\partial A}{\partial \mathbf{x}}\right] * \frac{\partial}{\partial \boldsymbol{\lambda}_1} (B_1 \boldsymbol{\lambda}_1) = \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1. \tag{12}$$

All matrices share the same typical structure: the critical information is in the first two entries of the first column. To evaluate A_2, the matrices we need to store are the Jacobian matrix A of the given system and its derivatives $\frac{\partial A}{\partial \mathbf{x}}$ and $\frac{\partial^2 A}{\partial \mathbf{x}^2}$. The role of the multipliers $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ in the evaluation is strictly linear, they occur only in matrix-vector products.

For $k = 3$, the evaluation of

$$A_3(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) = \left[\begin{array}{cc} A_2 & \mathbf{0} \\ \left[\frac{\partial A_2}{\partial \mathbf{x}} \ \frac{\partial A_2}{\partial \boldsymbol{\lambda}_1} \ \frac{\partial A_2}{\partial \boldsymbol{\lambda}_2}\right] B_3 \boldsymbol{\lambda}_3 & A_2 B_3 \\ \mathbf{0} & \mathbf{h}_3 \end{array} \right] \tag{13}$$

requires the evaluation of $A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$ – which we considered above – and its partial derivatives $\frac{\partial A_2}{\partial \mathbf{x}}$, $\frac{\partial A_2}{\partial \boldsymbol{\lambda}_1}$, $\frac{\partial A_2}{\partial \boldsymbol{\lambda}_2}$ respectively listed below:

$$\frac{\partial A_2}{\partial \mathbf{x}} = \left[\begin{array}{cc} \frac{\partial A_1}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A_1}{\partial \mathbf{x}^2} \ \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \boldsymbol{\lambda}_1}\right] B_2 \boldsymbol{\lambda}_2 & \left[\frac{\partial A_1}{\partial \mathbf{x}}\right] B_2 \\ \mathbf{0} & \mathbf{0} \end{array} \right], \tag{14}$$

$$\frac{\partial A_2}{\partial \boldsymbol{\lambda}_1} = \left[\begin{array}{cc} \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} & \mathbf{0} \\ \left[\frac{\partial^2 A_1}{\partial \boldsymbol{\lambda}_1 \partial \mathbf{x}} \ \frac{\partial^2 A_1}{\partial \boldsymbol{\lambda}_1^2}\right] B_2 \boldsymbol{\lambda}_2 & \left[\frac{\partial A_1}{\partial \boldsymbol{\lambda}_1}\right] B_2 \\ \mathbf{0} & \mathbf{0} \end{array} \right], \tag{15}$$

and

$$\frac{\partial A_2}{\partial \boldsymbol{\lambda}_2} = \left[\begin{array}{cc} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} \; \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \right] * B_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right]. \tag{16}$$

As the multipliers occur linearly, we have that $\frac{\partial^2 A_1}{\partial \boldsymbol{\lambda}_1^2} = \mathbf{0}$. Observing $\frac{\partial^2 A_1}{\partial \boldsymbol{\lambda}_1 \partial \mathbf{x}} = \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \boldsymbol{\lambda}_1}$, we have two more matrices to compute:

$$\frac{\partial^2 A_1}{\partial \mathbf{x}^2} = \left[\begin{array}{cc} \frac{\partial^2 A}{\partial \mathbf{x}^2} & \mathbf{0} \\ \left[\frac{\partial^3 A}{\partial \mathbf{x}^3} \right] B_1 \boldsymbol{\lambda}_1 & \left[\frac{\partial^2 A}{\partial \mathbf{x}^2} \right] B_1 \\ \mathbf{0} & \mathbf{0} \end{array} \right] \text{ and } \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \boldsymbol{\lambda}_1} = \left[\begin{array}{cc} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2} \right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right]. \tag{17}$$

Thus the evaluation of $A_3(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3)$ requires $\left[A \; \frac{\partial A}{\partial \mathbf{x}} \; \frac{\partial^2 A}{\partial \mathbf{x}^2} \; \frac{\partial^3 A}{\partial \mathbf{x}^3} \right]$. The partial derivatives with respect to the multiplier variables in $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_3$ do not need to be computed explicitly.

Just as a tree is a natural data structure to store the derivatives of A, a tree is used to represent the deflation matrices. For memory efficiency, the same node should only be stored once and a remember table is used in the recursive creation of the tree, which is actually a directed acyclic graph, see Fig. 1.

$$\begin{tikzcd}
 & & A_3(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & & \\
A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \arrow[d] \arrow[dr] \arrow[drr] & \frac{\partial A_2}{\partial \mathbf{x}} \arrow[drr] \arrow[drrr] & \frac{\partial A_2}{\partial \boldsymbol{\lambda}_1} \arrow[d] \arrow[drr] & \frac{\partial A_2}{\partial \boldsymbol{\lambda}_2} \arrow[dll] \arrow[dl] & \\
A_1(\mathbf{x}, \boldsymbol{\lambda}_1) \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \mathbf{x}} \arrow[d] \arrow[dr] & \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \arrow[dl] & \frac{\partial^2 A_1}{\partial \mathbf{x}^2} \arrow[dl] \arrow[d] & \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \boldsymbol{\lambda}_1} \arrow[dll] \\
A(\mathbf{x}) & \frac{\partial A}{\partial \mathbf{x}} & \frac{\partial^2 A}{\partial \mathbf{x}^2} & \frac{\partial^3 A}{\partial \mathbf{x}^3} &
\end{tikzcd}$$

Figure 1. Directed acyclic graph illustrating the dependency relations in the evaluation of A_3.

The graph in Fig. 1 has 14 nodes. If we perform the deflation three times on a 10-by-10 system (10 equations in 10 variables), each time with the corank 1 (worst case scenario: maximal number of multipliers), then A_3 would be a 87-by-80 matrix and the graph of evaluated matrices would occupy about 347Kb (8 bytes for one complex number of two doubles). In case the rank would always be zero (best case: only one multiplier in each case), then A_3 would be a 87-by-13 matrix and the graph of evaluated matrices would occupy about 257Kb.

Table 1. Growth of the number of distinct derivative operators, as nodes in a directed acyclic graph, for increasing deflations k. The case $k = 3$ is displayed in Fig. 1.

k	1	2	3	4	5	6	7	8	9	10
#nodes	3	7	14	26	46	79	133	221	364	596

The size of the graph grows modestly, see Table 1. For example, for $k = 10$, we have 596 nodes. On the other hand, we must be careful to remove evaluated matrices when no longer needed. For $k = 10$ on a 10-by-10 system we would need at least 206Mb in the best case and 8Gb if the corank is always one.

We conclude this section with an anecdotal reference to actual timings, done on a 2.4 Ghz linux workstation, with the system cyclic 9-roots, illustrating that even for a modest number of deflations, it pays off to use a directed acyclic graph. Only to show the benefits of an efficient evaluation, we deflate twice, assuming the corank is one each time, and end up with a 39-by-36 Jacobian matrix. It takes only 30 milliseconds to evaluate this matrix using the pre-calculated directed acyclic graph, which takes less than a second to set up. Computing the symbolic representation of this Jacobian matrix in a straightforward manner and evaluating it takes 25 minutes and 40 seconds!

5. The Deflation Algorithm in *PHCmaple*

Our Maple package *PHCmaple* [7] provides a convenient interface to the functions of PHCpack. The interface exploits the benefits of linking computer algebra with numerical software. *PHCmaple* is a first step in a larger project aimed at integration of a numerical solver in a computer algebra system.

Below we give an example of using the *PHCmaple* function `deflationStep` to deflate the system of equations `Ojika1` (copied from [10] and member of our benchmark collection in [8]):

$$\left\{ \begin{array}{rcl} x^2 + y - 3 & = & 0 \\ x + 0.125y^2 - 1.5 & = & 0. \end{array} \right. \tag{18}$$

The system has two isolated solutions: $(x, y) = (-3, -6)$ which is regular and $(x, y) = (1, 2)$ which is multiple. The function `deflationStep` takes a system and a list of solutions approximations as parameters, constructs the deflated systems for all multiple roots in the list, and returns the list of lists of solutions. The latter are being grouped according to the rank of the Jacobian at the points, since for the points with the same rank a common deflated system may be constructed.

```
> T := makeSystem([x,y],[],[x^2+y-3, x+0.125*y^2-1.5]):
> sols := solve(T):
> printSolutions(T,sols);
   (1) [x = 1.0000+.44913e-5*I, y = 2.0000-.89826e-5*I]

   (2) [x = -3.0, y = -6.0]

   (3) [x = 1.0000+.60400e-5*I, y = 2.0000-.12080e-4*I]

   (4) [x = 1.0000-.38258e-5*I, y = 2.0000+.76515e-5*I]
```

To each group of points corresponds a table:

```
> l := map(i->table(i),deflationStep(sols,T)):
```

Solution (2) is simple (the rank of its Jacobian is full), the cluster of (1),(3),(4) approximates a multiple solution (rank = 1 < 2)

```
> map(g->[rank=g["rank"],num_points=nops(g["points"])],l);
```

$$[[rank = 2,\ num\_points = 1],\ [rank = 1,\ num\_points = 3]]$$

The multipliers used in the deflation step are

```
> multipliers := l[2]["multipliers"];
```

$$multipliers := [\lambda_1,\ \lambda_2]$$

The deflated system is linear in the multipliers (the matrix below is a part of (1) for k=1)

```
> DT := l[2]["deflated system"]:
> eqns :=
> map(p->p=0,DT:-polys[nops(T:-polys)+1..nops(DT:-polys)]):
> matrix(2,1,[A[0]*B[1],h[1]])=evalf[3](linalg[genmatrix](eqns,
> multipliers, b));
```

$$\begin{bmatrix} A_0\,B_1 \\ h_1 \end{bmatrix} = \begin{bmatrix} -1.51\,x - 1.31\,I\,x + 0.786 - 0.618\,I & 0.911\,x + 1.78\,I\,x - 0.0562 + 0.998\,I \\ 0.197\,y - 0.154\,I\,y - 0.755 - 0.656\,I & -0.0140\,y + 0.250\,I\,y + 0.455 + 0.890\,I \\ -0.681 + 0.732\,I & -0.115 - 0.993\,I \end{bmatrix}$$

```
> DTmu := subsVariables
     (DT,[seq(lambda[i]=mu[i],i=1..l[2]["rank"]+1)]):
> DTmu:-vars;
```

$$[x,\ y,\ \mu_1,\ \mu_2]$$

```
> g := table(deflationStep(l[2]["points"],DTmu)[1]):
> printSolutions(g["deflated system"],g["points"]);
(1) [x = 1.0-.87360e-15*I, y = 2.0+.18332e-14*I,
   mu[1] = -1.2402-.53612e-1*I, mu[2] = -.87951+.15347e-1*I,
   lambda[1] = -.85183-.21804*I, lambda[2] = -.91795+.57730*I,
   lambda[3] = .64096-.77868*I, lambda[4] = -.21656-.75758*I]
```

Get the multiplicity of the multiple solution:

```
> mult_solution := g["points"][1]: mult_solution:-mult;
```

$$3$$

Compare conditioning to that of the original approximation.

```
> mult_solution:-rco , sols[1]:-rco;
```

$$0.02542, 0.9301\,10^{-11}$$

Above we have seen that the powerful symbolic-numeric capabilities of Maple are useful for presenting the original and deflated systems in a compact, non-expanded form. In the future we plan to represent the sequence of consecutive deflations simply by the sequences of matrices and vectors A_k, B_k, $\mathbf{h}_k$. Combined with the approach presented here, this would lead to a more efficient evaluation, as well as provide a better control over the deflation process to the user.

References

[1] D. J. Bates, C. Peterson and A. J. Sommese. A numerical-symbolic algorithm for computing the multiplicity of a component of an algebraic set. *J. Complexity* 22(4): 475–489, 2006.

[2] B. H. Dayton and Z. Zeng. Computing the multiplicity structure in solving polynomial systems. In M. Kauers, editor, *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, pages 116–123. ACM, 2005.

[3] R. D. Fierro and P. C. Hansen. Utv expansion pack: Special-purpose rank-revealing algorithms. *Numerical Algorithms* 40(1): 47–66, 2005.

[4] P. Kunkel. Efficient computation of singular points. *IMA J. Numer. Anal.* 9: 421–433, 1989.

[5] P. Kunkel. A tree-based analysis of a family of augmented systems for the computation of singular points. *IMA J. Numer. Anal.* 16: 501–527, 1996.

[6] G. Lecerf. Quadratic Newton iteration for systems with multiplicity. *Found. Comput. Math.* 2: 247–293, 2002.

[7] A. Leykin and J. Verschelde. PHCmaple: A Maple interface to the numerical homotopy algorithms in PHCpack. In Q.-N. Tran, editor, *Proceedings of the Tenth International Conference on Applications of Computer Algebra* (ACA'2004), pages 139–147, 2004.

[8] A. Leykin, J. Verschelde and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoretical Computer Science* 359(1-3): 111–122, 2006.

[9] T. Y. Li and Z. Zeng. A rank-revealing method with updating, downdating and applications. *SIAM J. Matrix Anal. Appl.* 26(4): 918–946, 2005.

[10] T. Ojika. Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations. *J. Math. Anal. Appl.* 123, 199–221, 1987.

[11] T. Ojika, S. Watanabe and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *J. Math. Anal. Appl.* 96: 463–479, 1983.

[12] H. J. Stetter. *Numerical Polynomial Algebra.* SIAM, 2004.

[13] H. J. Stetter and G. T. Thallinger. Singular systems of polynomials. In O. Gloor, editor, *Proceedings of ISSAC 1998*, pages 9–16. ACM, 1998.

[14] A. J. Sommese, J. Verschelde and C. W. Wampler. Numerical irreducible decomposition using PHCpack. In M. Joswig and N. Takayama, editors, *Algebra, Geometry, and Software Systems*, pages 109–130. Springer-Verlag, 2003.

[15] A. J. Sommese and C. W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science.* World Scientific, 2005.

[16] G. T. Thallinger. *Zero Behavior in Perturbed Systems of Polynomial Equations.* PhD Thesis, Tech. Univ. Vienna, 1998.

[17] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software* 25(2): 251–276, 1999. Software available at `http://www.math.uic.edu/~jan`.

[18] J. Verschelde and A. Zhao. Newton's method with deflation for isolated singularities. Poster presented at ISSAC'04, 6 July 2004, Santander, Spain. Available at `http://www.math.uic.edu/~jan/Talks/poster.pdf`.

Anton Leykin, Jan Verschelde and Ailing Zhao
University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
851 South Morgan (M/C 249), Chicago, IL 60607-7045, USA
e-mail: `leykin@math.uic.edu, URL: www.math.uic.edu/~leykin`
`jan@math.uic.edu, URL: www.math.uic.edu/~jan`
`azhao1@math.uic.edu, URL: www.math.uic.edu/~azhao1`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 279–298
© 2007 Birkhäuser Verlag Basel/Switzerland

On Approximate Linearized Triangular Decompositions

Marc Moreno Maza, Greg J. Reid, Robin Scott and Wenyuan Wu

Abstract. In this paper, we describe progress on the development of algorithms for triangular decomposition of approximate systems.

We begin with the treatment of linear, homogeneous systems with positive-dimensional solution spaces, and approximate coefficients. We use the Singular Value Decomposition to decompose such systems into a stable form, and discuss condition numbers for approximate triangular decompositions. Results from the linear case are used as the foundation of a discussion on the fully nonlinear case. We introduce linearized triangular sets, and show that we can obtain useful stability information about sets corresponding to different variable orderings. Examples are provided, experiments are described, and connections with the works of Sommese, Verschelde, and Wampler are made.

Mathematics Subject Classification (2000). Primary 13P10; Secondary 65L07.

Keywords. Symbolic-numeric computation, triangular decomposition, singular value decomposition, linear system.

1. Introduction

Systems of polynomial equations frequently arise in applications, and it is often of interest to determine their triangular forms. Such representations enable the expression of some of the variables as functions of the remaining ("free") variables. Already, methods exist which are designed to compute triangular sets for exact systems whose varieties are of arbitrary dimension [43, 22, 28, 42]. However, for real world problems, the systems under consideration frequently have approximate coefficients that are inferred from experimental data. This means that the stability of these triangular representations is a valid concern. To be more clear, we are not merely concerned with the sensitivity of the triangular representation to perturbations in the original data, but we also want the solution set to be stable under small changes in values taken by the free variables.

This work is supported by NSERC, MITACS, and Maplesoft, Canada.

In another paper, *On Approximate Triangular Decompositions I: Dimension Zero* [29], we gave a detailed treatment of approximate triangular decomposition for systems with finitely many roots (zero-dimensional systems). That work follows the equiprojectable decomposition presented in [9] and also makes use of the interpolation formulas recently proposed by Dahan and Schost [12]. In this second paper, we study the simplest class of positive dimensional systems: linear homogeneous systems.

It is true that Gaussian elimination, with respect to a given variable ordering, will transform any exact linear system provided as input into a triangular solved form. Solutions are parameterized by free variables which are lower in the ordering than the remaining variables. However, in the case of approximate systems, neither replacement of floating point numbers with rational numbers nor use of Gaussian elimination with full pivoting can be guaranteed to give orderings which are ideal (see the standard text [21] for a discussion of these issues). Furthermore, such methods may lead to approximate triangular representations which are practically unstable, even in the case of exact systems.

The key idea for the portion of the current paper which deals with linear, homogeneous systems, is to use stable methods from Numerical Linear Algebra. Specifically we use the Singular Value Decomposition (SVD) to determine whether a stable approximate triangular set exists for some variable ordering. We will show that such a stable set and ordering always exists, but that a given ordering may lead to an unstable representation. Furthermore, an interpretation of this set is given in terms of an exact solution to some nearby (homogeneous) system.

From there we will further explore some local structure of nonlinear problems with Linearized Approximate Triangular Decompositions. Applying results from the treatment of linear problems to the linearization of nonlinear systems, variable orderings for locally-stable approximate triangular sets can be determined. To do this, we use the homotopy continuation methods of PHCpack [41] to generate generic points on each irreducible component of a given nonlinear polynomial system [32, 34, 35, 33]. A collection of certain results on the decomposition of non-linear systems is also provided here. For example, for n variables, the interpolation methods of Sommese, Verschelde, and Wampler give approximate triangular representations of the $(n-1)$-dimensional components.

The above results, together with certain results using the equiprojectable decomposition (presented in our related work [29]), form an accessible bridge to the study of the fully non-linear case. This will be described in a forthcoming work.

2. Approximate Triangular Sets for Positive Dimensional Linear Systems

In this section we discuss triangular representations for linear systems. The foundation of our treatment is the Singular Value Decomposition, and techniques that have now become standard in Numerical Linear Algebra [21, 39].

2.1. The Singular Value Decomposition and Variable Orderings

Suppose we are given a system of m linear homogeneous equations in n variables, and that the solution space of the system has some positive dimension d. We wish to express the solution with $n-d$ variables in terms of d free variables.

Indeed, for reasons which will become clear, we worry that Gaussian elimination might produce an unstable (triangular) representation of the solution set. Instead of Gaussian elimination, our main tool will be the Singular Value Decomposition (SVD) of Numerical Linear Algebra.

Given $A \in \mathbb{F}^{m\times n}$ where $\mathbb{F}$ is $\mathbb{R}$ or $\mathbb{C}$ one can compute the SVD [21, 39]:

$$A = U\Sigma V^t.$$

Here, Σ is a diagonal matrix with the same dimensions as A, both $U \in \mathbb{F}^{m\times m}$ and $V \in \mathbb{F}^{n\times n}$ are orthogonal matrices, and V^t denotes the transpose of V if $\mathbb{F}=\mathbb{R}$ or the complex conjugate transpose if $\mathbb{F}=\mathbb{C}$. If the rank of A is r, then the last $n-r$ columns of V form a basis for Null(A). The diagonal elements of Σ are real, and are called the *singular values* of A. Often, it is enough to take the number of nonzero singular values as the numerical rank of A. For later purposes, it is useful to mention that the singular values are the square roots of the eigenvalues of matrices A^tA and AA^t (where the transpose is the Hermitian transpose if $\mathbb{F}=\mathbb{C}$).

Let $P \in \mathbb{F}^{n\times d}$ be a matrix whose columns form a basis for Null(A). In particular, we can let P be composed of the last $n-r$ columns of V from the SVD of A and define $d := n-r$. Then

$$\begin{aligned}\mathbf{x} &= \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \ldots + \alpha_d\mathbf{p}_d \\ &= P\boldsymbol{\alpha}, \end{aligned} \tag{2.1}$$

with $\boldsymbol{\alpha} \in \mathbb{F}^d$. Furthermore, the matrix P is stable with respect to changes in the (approximate) coefficients of A. Indeed, adjoining $\alpha_1, \ldots, \alpha_d$ as new indeterminates we simply have the following:

Remark 2.1. Equation (2.1) is a (normalized) triangular set in any ordering ranking the variables $\{x_1, \ldots, x_n\}$ greater than the variables $\{\alpha_1, \ldots, \alpha_d\}$.

Definition 2.2. A triangular representation of the form

$$\mathbf{x}_{\text{non-free}} = M\mathbf{x}_{\text{free}}, \tag{2.2}$$

where M is a matrix with entries in $\mathbb{F}$, is *practically stable* if the condition number of M is reasonable. Thus, small changes in the free variables of a practically stable solution will not induce huge changes in the non-free variables.

Remark 2.3. The triangular set (2.1) is practically stable. We write

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \|P\|\|P^\dagger\|\frac{\|\delta\boldsymbol{\alpha}\|}{\|\boldsymbol{\alpha}\|} \tag{2.3}$$

to estimate the sensitivity of the solution $\mathbf{x}$ (the non-free variables) to small perturbations of $\boldsymbol{\alpha}$ (the free variables), given the matrix P. Since P is not a square matrix, the usual inverse has been replaced with the pseudo-inverse $P^\dagger$ (see Remark 2.8 for more details), and the condition number of the triangular solution (2.1) can still be expressed in terms of the singular values of P. Since the columns of P are orthogonal, the matrix P^tP is the identity matrix, whose eigenvalues are all equal to 1. Thus, every singular value of P is equal to 1, and so its condition number

$$\|P\|_2\|P^\dagger\|_2 = \sigma_1\frac{1}{\sigma_d} = 1. \tag{2.4}$$

This result is independent of the condition number of the given matrix A.

We will now further our development by investigating approximate linear triangular sets with respect to orderings of the variables $\{x_1, \ldots, x_n\}$ alone. It is natural to ask why we should investigate such sets, given that, by Remark 2.1, the SVD already directly yields the triangular set (2.1). One reason is that in applications we are often interested in expressing some variables in terms of others. Another, more mathematical, reason is that we wish to use the linear case as our guide in the study of the positive dimensional nonlinear case where such orderings on the variables $\{x_1, \ldots, x_n\}$ are often used.

To obtain triangular sets in the variables $\{x_1, \ldots, x_n\}$ alone, we need to eliminate the variables $\{\alpha_1, \ldots, \alpha_d\}$ in (2.1) in favor of d free variables suitably chosen from $\{x_1, \ldots, x_n\}$. The remaining $n - d$ variables will then be expressed as functions of these free variables instead of the α's. If we can find an invertible sub-matrix Q of P, then

$$\mathbf{x}_{\text{free}} = Q\boldsymbol{\alpha}, \tag{2.5}$$

and consequently

$$\boldsymbol{\alpha} = Q^{-1}\mathbf{x}_{\text{free}}. \tag{2.6}$$

Denoting by R the matrix composed of the $n-d$ rows of P that do not appear in Q, the remaining variables may be expressed as

$$\begin{aligned} \mathbf{x}_{\text{non-free}} &= R\boldsymbol{\alpha} \\ &= RQ^{-1}\mathbf{x}_{\text{free}}. \end{aligned} \tag{2.7}$$

Theorem 2.4. *For every $A \in \mathbb{F}^{m\times n}$ with dimension $d = n - r \neq 0$, there is always an ordering of the variables $\{x_1, x_2, \ldots, x_n\}$ which can be used to produce a stable triangular representation of the linear system $A\mathbf{x} = \mathbf{0}$.*

Proof. An invertible $d \times d$ sub-matrix of P will always exist since P has rank d. Moreover, the SVD guarantees that the computation of P is stable with respect to small perturbations in the coefficients of A. Thus, we will always have an ordering: $\{\mathbf{x}_{\text{non-free}}\} \succ \{\mathbf{x}_{\text{free}}\}$ with which we can write the solutions to $A\mathbf{x} = \mathbf{0}$ in the form of (2.7). (The practical stability of (2.7) will be discussed in Sect. 2.3.) $\square$

2.2. Backward Error Analysis for Positive Dimensional Linear Homogeneous Systems

If the matrix A, with a positive dimensional null space, consists of entries which are approximate, a numerical solution to $A\mathbf{x} = 0$ will not likely solve this given problem exactly. However, it is standard to say that this solution exactly solves some nearby problem. We will present some such results on backward error analysis for the triangular solutions described in this paper. (For some interesting works on backward error analysis in both very general and applied settings, see [38] and references therein.)

If the solution that we find is $\mathbf{x}^*$, it does not make sense to say that we have solved $A\mathbf{x}^* = \mathbf{e}$ exactly for some nonzero vector of constants $\mathbf{e}$. Here are two reasons why:

(1) The residual is really of the form $\mathbf{e(x)}$, a vector which depends on all of the variables $\{x_1, x_2, \ldots, x_n\}$;
(2) Since our problem is homogeneous, the nearby problem which we have solved should also be homogeneous.

So, we want to say that we have solved $\tilde{A}\mathbf{x}^* = 0$ where $\mathbf{x}^* = X\mathbf{x}$, $X \in \mathbb{F}^{n\times n}$, and $\|A - \tilde{A}\|$ is small. In the context of the development in Sect. 2.1, we can find a *triangular* solution of the form $\mathbf{x}_{\text{non-free}} = RQ^{-1}\mathbf{x}_{\text{free}}$. So the problem that we have solved can be written as

$$A\left[\frac{RQ^{-1}}{\tilde{I}}\right]\mathbf{x}_{\text{free}} = E\mathbf{x}_{\text{free}}. \tag{2.8}$$

Here the columns of A have simply been rearranged so that $\left[\frac{RQ^{-1}}{\tilde{I}}\right]$ may be used to denote the $n \times d$ matrix with first (top) $n-d$ rows formed by the $n-d$ rows of RQ^{-1}, and the remaining (lowest) d rows are a suitable permutation of the identity matrix. The right hand side of (2.8) is non-zero because we are unlikely to find the exact solution to the original (approximate) problem. Presumably, there is some perturbation matrix δA with which we can write

$$(A - \delta A)\left[\frac{RQ^{-1}}{\tilde{I}}\right]\mathbf{x}_{\text{free}} = \mathbf{0}. \tag{2.9}$$

With Equations (2.8) and (2.9), we can see that

$$\delta A\left[\frac{RQ^{-1}}{\tilde{I}}\right] = E. \tag{2.10}$$

Proposition 2.5. *Given an approximate, linear, homogeneous equation $A\mathbf{x} = \mathbf{0}$, and a triangular representation for its solutions, there exists a perturbation to the coefficients of A so that this triangular solution exactly solves the perturbed system $(A - \delta A)$. Furthermore, we have $\|\delta A\| \leqslant \|E\|$.*

Proof. Transposing (2.10) gives

$$\left[RQ^{-1}|\tilde{I}\right]\delta A^t = E^t, \tag{2.11}$$

and we are free to choose the top $n-d$ rows of δA^t to be zero rows. The remaining d rows are then a permutation of E^t such that $\tilde{I}\tilde{E}^t = E$. With this choice, we have $\|\delta A\| = \|E\|$. There may also be alternative better choices of δA for which $\|\delta A\| < \|E\|$. $\square$

2.3. Some Stability Results: in Theory and in Practice

In Sect. 2.1, we had claimed that there is an invertible $d \times d$ sub-matrix of P. However, this is a theoretical result which, in the approximate case, is rather subtle.

If P was exact, it would then contain exactly d linearly independent rows and $n-d$ linearly dependent rows. Furthermore, any particular row is linearly dependent on the other rows if and only if it is able to be expressed as an exact linear combination of them. In our case, the entries of P are approximate, and it is often quite unlikely that any row is an exact linear combination of the others. Thus, we cannot speak of having exactly d linearly independent and $n-d$ dependent rows. However, we are still guaranteed a $d \times d$ sub-matrix of P which is invertible, but there will likely be more than one such sub-matrix. So, our goal is to select for our Q the "best" $d \times d$ sub-matrix of P.

Theorem 2.6. *Given a matrix $A = U\Sigma V^t \in \mathbb{F}^{m\times n}$ with rank r, the closest singular matrix to A has rank $r-1$ and can be constructed as $\tilde{A} = U\tilde{\Sigma}V^t$, where $\tilde{\Sigma}$ is equal to Σ with σ_r replaced by zero. Furthermore, we have $\|A-\tilde{A}\|_2 = \sigma_r$.*

Proof. See Trefethen and Bau [39] for an even more general result and proof. $\square$

Remark 2.7. The theorem about the approximation of a given matrix by another of lower rank is usually attributed to Eckart and Young [13], and is often called the Eckart-Young theorem. However, the theorem was first proved by Schmidt (1907) for integral operators, and was later generalized by Mirsky (1960) to all unitarily invariant norms.

This means that for any $d \times d$ sub-matrix Q with singular values $\{\sigma_1, \sigma_2, \ldots, \sigma_d\}$, there exists a singular matrix within distance σ_d of Q. In lieu of Theorem 2.6, our initial inclination was to claim that the "best" Q is the $d \times d$ sub-matrix of P which has the largest σ_d. Actually, the choice of the best Q is not quite that simple. It is true that we want a nonsingular Q, but, unfortunately, the Q which is farthest from singular may not give us the RQ^{-1} which is most practically stable.

Before going further, we wish to state some well-known results which will be used in an explanation of the conjecture which concludes this section.

Remark 2.8. It is well-known that for any given matrix $A = U\Sigma \mathrm{V}^t$, $\|A\|_2 = \sigma_1$. For any invertible matrix $A \in \mathbb{F}^{n\times n}$, we have $A^{-1} = V\Sigma^{-1}U$, and so the singular values of A^{-1} are $\{\frac{1}{\sigma_n}, \frac{1}{\sigma_{n-1}}, \ldots, \frac{1}{\sigma_1}\}$, and $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$. If A is not square, then its pseudo-inverse $A^\dagger = V\Sigma^\dagger U^t$ can be computed instead of the inverse. For our purposes, we regard A as having full rank, and then the diagonal elements of $\Sigma^\dagger$ are still $\{\frac{1}{\sigma_n}, \frac{1}{\sigma_{n-1}}, \ldots, \frac{1}{\sigma_1}\}$. See, for example, [21] for more information on the pseudo-inverse.

Remark 2.9. Without proof we state the known result that if M is any sub-matrix of a given matrix A, then $\|M\|_p \leqslant \|A\|_p$ for any matrix p-norm.

A stable solution will have continuous dependence on its input data. However, it may be very sensitive to changes in the values substituted for the free variables. The acceptable magnitude of the condition number will depend on the particular application that is involved, and will determine if the triangular representation is practically stable. We introduce the notation: σ_Q and σ_R for the last (d^{th}) singular values of Q and R respectively. Simply, using Remarks 2.8 and 2.9, and the fact that $\|P\|_2 = 1$ (shown in Remark 2.3), we have the following bound for the condition number of RQ^{-1}:

$$\begin{aligned}\mathrm{Cond}(RQ^{-1}) &= \|RQ^{-1}\|_2\|(RQ^{-1})^{-1}\|_2 \\ &\leqslant \|R\|_2\|Q^{-1}\|_2\|Q\|_2\|R^{\dagger}\|_2 \\ &\leqslant \|P\|_2\frac{1}{\sigma_Q}\|P\|_2\frac{1}{\sigma_R} \\ &= \frac{1}{\sigma_Q}\cdot\frac{1}{\sigma_R}. \qquad (2.12)\end{aligned}$$

By maximizing the product $\sigma_Q \cdot \sigma_R$, we will have minimized the bound on $\mathrm{Cond}(RQ^{-1})$. In our experiments, we have observed that choosing Q and R such that Q is non-singular, and $\sigma_Q \cdot \sigma_R$ is maximal, yielded a solution for which the condition number was minimal. Simply choosing Q such that it is the $d \times d$ sub-matrix which is farthest from singular does not guarantee the best solution (see the example in Sect. 3).

Conjecture 2.10. For every $A \in \mathbb{F}^{m\times n}$ with dimension $d = n - r \neq 0$, we can write $\mathbf{x}_{\text{non-free}} = RQ^{-1}\mathbf{x}_{\text{free}}$. By choosing Q non-singular and such that $\sigma_Q \cdot \sigma_R$ is maximal, we will have found ordered sets $\{\mathbf{x}_{\text{non-free}}\} \succ \{\mathbf{x}_{\text{free}}\}$ for which the stable triangular representation of this linear system is most practically stable.

2.4. Some Notes on Computation

To find an invertible sub-matrix Q of P, and corresponding sub-matrix R, is one challenging step in computing a practically stable triangular representation. It is considerably more difficult still to find the "best" combination of Q and R. An ideal algorithm would be one which fulfills the following two criteria:

(1) The output variable ordering is optimal;
(2) The method is efficient.

This is not a straight-forward problem. Of course there exists an algorithm which will fulfill criterion (1): simply use brute force to try all $\binom{n}{d}$ combinations of d rows taken from P. This is a sensible strategy for small problems, but the expense of trying to solve systems which admit large P is unsatisfactory. We also note that there may be several optimal choices (e.g. consider $x + y + z = 0$, where every ordering is optimal).

Employing randomness can give us (2), but not necessarily (1). We can randomly select d rows from P. For large P, we are not likely to select the best Q

(and R), but we can also expect not to obtain the worst combination. For an even better solution, one could repeat this a number of times, and take the best result. For further study, one would like to know more about the distribution of the σ_d for all $d \times d$ and $n - d \times d$ sub-matrices of P. Furthermore, thought on this strategy raises another question: if we are not guaranteed an optimal solution, what may be accepted as a "sufficient" solution?

3. A Linear Example

In this example, we use the 10×15 Hilbert Matrix, which has full rank and has as each entry $h_{i,j} = \frac{1}{i+j}$. We will denote this exact matrix by H_e and its floating point approximation by H_f.

First, we want to point out that the issue of stability is encountered in both numerical and exact approaches. The stability of a triangular representation for the solutions of an *exact* system may also be ordering-dependent. Suppose that we are given a system of linear, homogeneous polynomials $P = \{p_1, \ldots, p_m\} \in \mathbb{Q} \in [x_1, \ldots, x_n]$, and that we want a triangularization of the solutions to $P\mathbf{x} = 0$ in the form of (2.7). The coefficients in the solution may be computed exactly (say, using exact Gaussian elimination). However, the solutions to the system are still real. The substitution of real values for the free variables admits the possibility of practical instability.

Using Gaussian elimination to solve $H_e\mathbf{x} = \mathbf{0}$ with $\mathbf{x} = (x_1, \ldots, x_{15})$, we achieve the variable ordering $\{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\} \prec \{x_1, x_2, \ldots, x_9, x_{10}\}$. This is as expected as H_e has (exactly) full rank. The condition number of this (triangular) solution is $\approx 2.66 \times 10^5$. Using the SVD, we can check that this solution corresponds to one for which $\sigma_Q \approx 0.11 \times 10^{-2}$, so Q is not singular. Also, we have $\sigma_R \approx 0.35 \times 10^{-2}$.

Another suitable solution strategy for linear polynomial solving is to use Gaussian elimination on the floating point system. This, instead of doing exact computation, can yield a considerable speed up, at least, due to a decrease in required storage and intermediate computations. With partial pivoting, Gaussian elimination does not re-order the columns of a given dimension$-d$ matrix A. Thus, generally, a solution will be achieved for which the free variables will be those associated with the last d columns of A. Gaussian elimination with full pivoting, however, will exchange the order of the columns. Although this strategy may consistently yield solutions for which the residual is small, the triangular representation may be unstable, or at least not guaranteed to be optimal.

Applying Gaussian elimination with full pivoting to solve $H_f\mathbf{x} = 0$, yields the ordering $\{x_2, x_3, x_4, x_5, x_6\} \prec \{x_1, x_7, \ldots, x_{14}, x_{15}\}$, and $\|E\|_F = 6.82 \times 10^{-15}$. So, at first glance, we appear to have found a good solution. However, the approximate triangular form is practically unstable, with computed condition number $\approx 1.29 \times 10^8$. Again using the SVD, we find that this corresponds to selecting a Q for which $\sigma_Q \approx 5.97 \times 10^{-9}$, and R such that $\sigma_R \approx 0.79$.

The solution for which Q is farthest from singular, with $\sigma_Q \approx 0.55$, has $\mathbf{x}_{\text{free}} = \{x_7, x_9, x_{11}, x_{12}, x_{14}\}$. Here, $\text{Cond}(RQ^{-1}) \approx 11.70$, so that this is close to the optimal solution, below.

There is (at least) one choice for Q and R such that $\text{Cond}(RQ^{-1})$ is minimized at ≈ 7.64. We have found one such partitioning with $\sigma_Q \approx 0.46$, $\sigma_R \approx 0.24$, and corresponding to $\mathbf{x}_{\text{free}} = \{x_6, x_8, x_{10}, x_{12}, x_{14}\}$.

We have seen that, for the purpose of finding practically stable triangular representations, both Gaussian elimination on exact systems and Gaussian elimination with full pivoting on approximate systems may indeed provide a poor variable ordering. Finally, it is interesting to note that, given an efficient and reliable method of computing an optimal ordering, one might think about using it to find practically stable triangular solutions even for exact systems.

4. Approximate Linearized Triangular Decompositions and Numerical Algebraic Geometry

In Sect. 4.1 we introduce triangular sets, linearized about points on the variety of a polynomial system (linearized triangular sets). The approximate points about which we linearize are computed using the methods of Numerical Algebraic Geometry due to Sommese, Verschelde, and Wampler. We discuss some background on this material in Sect. 4.2.

4.1. Approximate Linearized Triangular Decompositions

In this section we will make use of the results from our study of the linear case to introduce Linearized Triangular Sets. Suppose that we are given a nonlinear polynomial system $p = \{p_1, p_2, \ldots, p_m\}$ in n variables $\mathbf{x} = (x_1, \ldots, x_n)$. Then the affine variety of the system over $\mathbb{C}$ is defined as

$$V(p) = \{x \in \mathbb{C}^n : p(x) = 0\}. \tag{4.1}$$

The key idea of this section is to linearize about some given nonsingular point $\mathbf{x}^0 = (x_1^0, \ldots, x_n^0)$ on the variety of the system and let $\mathbf{v} = (x_1 - x_1^0, \ldots, x_n - x_n^0)^t$. This will yield the linear system

$$\frac{\partial p_j}{\partial x_k}\left(\mathbf{x}^0\right)\mathbf{v} = \mathbf{0}. \tag{4.2}$$

If p generates a radical ideal, it is easy to show that this linearized system is the tangent space of $V(p)$ at $\mathbf{x}^0$ [7]. We know the tangent space has the same dimension as the variety at this point. To avoid rank deficiency, the condition number of the Jacobian matrix at $\mathbf{x}^0$ must be considered. An extremely large condition number will either mean that this point is a multiple root, or that the polynomial system is ill-conditioned.

It is natural to use this linearization to study the fully non-linear positive dimensional case.

4.2. Numerical Algebraic Geometry

In order to achieve the linearization above we need to determine points $\mathbf{x}^0 \in V(p)$. The tools we use to determine approximations of such so-called "witness" points are the homotopy continuation methods of Sommese, Verschelde, and Wampler [37, 33].

In [37], Sommese and Wampler outlined the development of a new field "Numerical Algebraic Geometry," which led to the development of homotopies to describe all irreducible (or the weaker equidimensional) components of the solution set of a polynomial system.

An *irreducible affine variety* V is an affine variety that can not be expressed as a finite union of proper sub-varieties. A well-known result is that that any affine variety can be expressed uniquely as a union of finitely many irreducible affine varieties. The dimension of these irreducible components can vary from 0 to $n-1$:

$$V(p) = Z = \bigcup_{i=0}^{n-1} Z_i = \bigcup_{i=0}^{n-1} \bigcup_{j \in \mathcal{I}_i} Z_{ij}, \tag{4.3}$$

where Z_i is the union of all i-dimensional components, Z_{ij} are the irreducible components and $\mathcal{I}_i$ are index sets with finitely many entries. See Sect. 5 for an example.

Definition 4.1. Given a polynomial system $p(x) = 0$ having a decomposition Z into irreducible components Z_{ij} or more weakly into equidimensional components Z_i in (4.3), a witness set W is a set of points of the form

$$W := \bigcup_{i=0}^{n-1} W_i = \bigcup_{i=0}^{n-1} \bigcup_{j \in \mathcal{I}_i} W_{ij}, \tag{4.4}$$

where

1. W_{ij} is a finite sub-set of Z_{ij} (i.e. $W_{ij} \subset Z_{ij}$),
2. W_{ij} contains no points from any other $Z_{k\ell}$ (i.e. $W_{ij} \cap Z_{k\ell} = \emptyset$ for $(i,j) \neq (k,\ell)$),
3. W_{ij} contains $\deg Z_{ij}$ points, each occurring ν_{ij} times for some integer $\nu_{ij} \geq \mu_{ij}$ where μ_{ij} is the multiplicity of Z_{ij} as an irreducible component of $p^{-1}(0)$. Moreover, if $\mu_{ij} = 1$ then $\nu_{ij} = 1$.

See Sect. 5 for an example. The W_i above are the witness sets for the equidimensional decomposition and W_{ij} are the witness sets for the irreducible decomposition. Approximate points for these can be determined by numerical homotopy continuation methods. It is important to note that the witness sets for an equidimensional decomposition can be computed more cheaply than that for the irreducible decomposition.

4.3. Results for Approximate Triangular Decomposition

Examples illustrating the results of this section can be found in Sect. 5.

Since the witness points computed for the zero-dimensional case each have the form $x_1 = a_1, x_2 = a_2, \dots, x_n = a_n$, it follows that:

Remark 4.2. For an exactly given input system of polynomials over $\mathbb{C}$, the (exact) witness point representation of the set of isolated points Z_0 is a collection of triangular representations over $\mathbb{C}$.

The homotopy continuation methods of [33] give a method for approximating all such isolated points, provided a small enough tolerance is used, which rely for their stability on Bernstein's theorem [1]. Certification that the tolerance is small enough for Newton's method using approximate arithmetic to be certified as converging to an exact solution of the original exact system requires use, for example, of Shub and Smale's γ theory [2]. We note that the case of approximate input systems is more involved, and is the subject of current research.

In our previous paper, we reassembled this collection of approximate triangular representations to give an approximate equiprojectable decomposition of the Z_0 (which is an equidimensional decomposition). That triangular decomposition can be regarded as an approximation of the decomposition that is obtained by exact methods.

For the positive $(n-1)$-dimensional case, the witness set characterization [33] can be regarded as an exact, probability 1, non-algorithmic construction using exact complex arithmetic. In addition an exact interpolation process can be given, leading us to:

Remark 4.3. For systems with n variables, each such interpolating polynomial for each irreducible component of dimension $n-1$ constitutes a triangular representation for that irreducible component. In the equi $(n-1)$-dimensional case the single interpolation polynomial is also a triangular representation.

The methods of Sommese, Verschelde, and Wampler, regarded exactly, generate enough generic points to exactly interpolate each such irreducible component by a *single* polynomial in the variables x_1, x_2, $\dots$, x_n. In addition, by these methods a single interpolation polynomial can be obtained for Z_{n-1}, ie. for the components of dimension $n-1$ in the equidimensional decomposition. In the case of approximate arithmetic, the zero-dimensional systems arising in the above process again require certification for convergence, and algorithmic certification (in the sense of Shub and Smale) is an interesting topic of current research.

The interpolation procedure applied in Remark 4.3 incrementally increases the interpolation degree from 1 until the minimum degree where interpolation is successful is obtained, and yields triangular representations which generate radical ideals .

From Remarks 4.2 and 4.3 we have:

Remark 4.4. Approximate triangular representations can be obtained for each irreducible (and each equidimensional) component of a bivariate system of polynomials.

An easy consequence of Remark 2.4 is:

Remark 4.5. Given $\mathbf{x}^0 \in V(p)$, an ordering of the variables can be determined so that there is a stable linearized triangular representation of the polynomial system about $\mathbf{x}^0$.

Such linearized triangular systems give information on the existence and construction of associated nonlinear triangular systems. Again we add the cautionary remarks that the above statements are true as exact statements of $\mathbb{C}$, and care must be taken in approximate counterparts of such statements.

5. An Example of Sommese, Verschelde and Wampler

Consider the system, which is used as an illustrative example by [33]:

$$p = \begin{bmatrix} (y-x^2)(x^2+y^2+z^2-1)(x-0.5) \\ (z-x^3)(x^2+y^2+z^2-1)(y-0.5) \\ (y-x^2)(z-x^3)(x^2+y^2+z^2-1)(z-0.5) \end{bmatrix} = 0. \tag{5.1}$$

In this illustrative example, it is easy to find the decomposition

$$V(p) = Z_2 \cup Z_1 \cup Z_0 = \{Z_{21}\} \cup \{Z_{11} \cup Z_{12} \cup Z_{13} \cup Z_{14}\} \cup \{Z_{01}\}, \tag{5.2}$$

where

1. Z_{21} is the sphere $x^2+y^2+z^2-1=0$,
2. Z_{11} is the line $(x=0.5, z=0.5^3)$,
3. Z_{12} is the line $(x=\sqrt{0.5}, y=0.5)$,
4. Z_{13} is the line $(x=-\sqrt{0.5}, y=0.5)$,
5. Z_{14} is the twisted cubic $(y-x^2=0, z-x^3=0)$,
6. Z_{01} is the point $(x=0.5, y=0.5, z=0.5)$.

For the example above, which is executed by the algorithm IrreducibleDecomposition in [33], the witness set W, the degrees d_{ij} and multiplicity bounds ν_{ij} are

$$W = W_2 \cup W_1 \cup W_0 = \{W_{21}\} \cup \{W_{11} \cup W_{12} \cup W_{13} \cup W_{14}\} \cup \{W_{01}\}, \tag{5.3}$$

where

1. W_{21} contains 2 points, $d_{21}=2$ and $\nu_{21}=1$,
2. W_{11} contains 1 point, $d_{11}=1$ and $\nu_{11}=1$,
3. W_{12} contains 1 point, $d_{12}=1$ and $\nu_{12}=1$,
4. W_{13} contains 1 point, $d_{13}=1$ and $\nu_{13}=1$,
5. W_{14} contains 3 points, $d_{14}=3$ and $\nu_{14}=1$,
6. W_{01} is a non-singular point.

See Sect. 7.2 of [33] for the execution summary of this example.

Consider the system p from (5.1), with decomposition (5.2), and the description of the witness points and their degrees given above. As previously noted, the possible dimensions of the irreducible components are 0, 1, and $n-1=2$.

Zero-Dimensional Components: Using the methods of [33] yields one single isolated point of degree 1: $x=0.5$, $y=0.5$, $z=0.5$. This, of course, is a triangular set (and is also equiprojectable).

One-Dimensional Components: The methods of [33] predict 3 one-dimensional degree 1 (linear) components, and 1 one-dimensional degree 3 component. Immediately, the linearized triangular set method of Sect. 4.1 can be applied to approximate the linear one-dimensional components. Here is a sketch of the details.

First, from [33], we can determine an approximate generic point, $\mathbf{x}^0$, on each linear component: $\mathbf{a}\in Z_{11}$, $\mathbf{b}\in Z_{12}$, $\mathbf{c}\in Z_{13}$. Then, we can make use of the SVD to achieve linearized triangular representations of the form

$$\mathbf{x}=\mathbf{x}^0+\alpha\mathbf{w},\qquad \alpha\in\mathbb{C}, \tag{5.4}$$

where $\mathbf{x}^0\in\{\mathbf{a},\mathbf{b},\mathbf{c}\}$ and $\mathbf{w}\in\{\mathbf{w}_a,\mathbf{w}_b,\mathbf{w}_c\}$ are the corresponding basis vectors given by the SVD. Here, α is regarded as an additional indeterminate with $\{x,y,z\}\succ\alpha$. Recalling (2.7), if $w_k\neq 0$ we can choose x_k as a free variable and rewrite (5.4) as

$$x_j=x_j^0+\frac{w_j}{w_k}\left(x_k-x_k^0\right), \tag{5.5}$$

where all other x_j $(j\neq k)$ are the two non-free variables.

We used for $\mathbf{x}^0$ (respectively for each of the 3 linear 1-dimensional components):

$$\begin{aligned}
\tilde{\mathbf{a}} &\approx \left[0.50+2.08\times10^{-11}i, 1.00+1.64\times10^{-11}i, 0.125-7.13\times10^{-11}i\right]^t,\\
\tilde{\mathbf{b}} &\approx \left[0.71+4.10\times10^{-11}i, 0.50-9.30\times10^{-11}i, 1.00+2.92\times10^{-11}i\right]^t,\\
\tilde{\mathbf{c}} &\approx \left[-0.71-9.68\times10^{-11}i, 0.50-4.28\times10^{-11}i, 1.00-8.51\times10^{-11}i\right]^t.
\end{aligned}$$

The corresponding vectors $\mathbf{w}$ obtained by applying the SVD to the linearized systems about $\mathbf{x}^0$ are

$$\begin{aligned}
\mathbf{w}_a &\approx [-2.50\times10^{-10}-0.i, 0.68+0.74i, 3.37\times10^{-10}-6.32\times10^{-10}i]^t,\\
\mathbf{w}_b &\approx [-5.22\times10^{-10}+0.i, 1.73\times10^{-10}+3.54\times10^{-10}i, -0.85+0.53i]^t,\\
\mathbf{w}_c &\approx [1.51\times10^{-10}+0.i, 1.39\times10^{-11}+3.64\times10^{-10}i, -0.73-0.69i]^t,
\end{aligned}$$

where we have rounded the above results to two decimal places (and $0.i$ is used to denote floating point zero). The next step is to find the stable orderings of $\{x,y,z\}$ on each linear component.

Z_{11}**:** For the one-dimensional case, we do want to find the invertible $d\times d$ sub-matrix Q of $\mathbf{w}_a$ which has greatest d^{th} singular value. Since here we have $d=1$, the singular values of each of the three 1×1 sub-matrix of $\mathbf{w}_a$ are simply the magnitudes of each complex entry itself. This means that we are looking for the entry of $\mathbf{w}_a$ with greatest magnitude, and this will tell us which of $\{x, y, z\}$ will be the best free-variable. We find a single stable choice:

$$\begin{aligned} \mathbf{x}_{\text{free}} &= \{y\}, \\ \mathbf{x}_{\text{non-free}} &= \{x, z\}, \end{aligned}$$

which corresponds to either of the orderings: $y \prec x \prec z$, or $y \prec z \prec x$.

Geometrically, this can be interpreted as follows: the linear component Z_{11} is exactly a line perpendicular to the xz-plane. Because the numerical approximation to this component contains errors (due in part to the inexact point $\tilde{\mathbf{a}}$), it is a line which is not exactly perpendicular to the xz-plane. Solving for y, z in terms of x, or x, y in terms of z, are unstable choices. A small change in x would then cause a change in y roughly on the order of 10^9 (in the first case), or a change in z would cause a change in y which would also be around the order of 10^9 (in the second case).

The (only) stable approximate linearized triangular representation is

$$\begin{aligned} x &= \tilde{a}_x - (1.69 \times 10^{-10} - 1.84 \times 10^{-10} i)(y - \tilde{a}_y), \\ z &= \tilde{a}_z - (2.37 \times 10^{-10} + 6.76 \times 10^{-10} i)(y - \tilde{a}_y). \end{aligned}$$

Z_{12}**:** As for Z_{11}, we find just one stable choice of free variables:

$$\begin{aligned} \mathbf{x}_{\text{free}} &= \{z\}, \\ \mathbf{x}_{\text{non-free}} &= \{x, y\}, \end{aligned}$$

which corresponds to either of the orderings: $z \prec x \prec y$, or $z \prec y \prec x$. This corresponds to the approximate solution

$$\begin{aligned} x &= \tilde{b}_x + (4.43 \times 10^{-10} + 2.76 \times 10^{-10} i)(z - \tilde{b}_z), \\ y &= \tilde{b}_y + (4.07 \times 10^{-11} - 3.92 \times 10^{-10} i)(z - \tilde{b}_z). \end{aligned}$$

Z_{13}**:** Here the stable choice of free variables is

$$\begin{aligned} \mathbf{x}_{\text{free}} &= \{z\}, \\ \mathbf{x}_{\text{non-free}} &= \{x, y\}, \end{aligned}$$

which corresponds to either of the orderings: $z \prec x \prec y$, or $z \prec y \prec x$. The stable approximate solution is

$$\begin{aligned} x &= \tilde{c}_x - (1.10 \times 10^{-10} - 1.04 \times 10^{-10} i)(z - \tilde{c}_z), \\ y &= \tilde{c}_y - (2.60 \times 10^{-10} + 2.55 \times 10^{-10} i)(z - \tilde{c}_z), \end{aligned}$$

It is both interesting and important to note that for exact triangular decomposition, one can set a single order for the entire computation. However, here there is no single stable choice of $\{\mathbf{x}_{\text{non-free}}\}$ and $\{\mathbf{x}_{\text{free}}\}$ which covers all three cases (Z_{11}, Z_{12}, and Z_{13}).

Z_{14} (*Twisted Cubic*): Consider the linearization of p about the (random) point $\mathbf{x}^0$ on the twisted cubic given by

$$\tilde{\mathbf{d}} \approx [0.50 + 4.16 \times 10^{-11} i, 0.25 + 3.41 \times 10^{-11} i, 0.125 + 9.63 \times 10^{-11} i]^t.$$

Applying the SVD to the linearization of the system about $\mathbf{x}^0 = \mathbf{d}$ yields $\mathbf{w}$ given by

$$\mathbf{w}_d \approx [0.48 + 0.i, 0.80 + 0.10i, 0.36 - 4.30 \times 10^{-10}i]^t.$$

Then the method above yields the most stable choice

$$\begin{aligned} \mathbf{x}_{\text{free}} &= \{y\}, \\ \mathbf{x}_{\text{non-free}} &= \{x, z\}, \end{aligned}$$

which corresponds to either $y \prec x \prec z$, or $y \prec z \prec x$. We obtain the following linearized triangular decomposition:

$$\begin{aligned} x &= \tilde{d}_x - (5.90 \times 10^{-1} - 0.76 \times 10^{-1}i)(y - \tilde{d}_y), \\ z &= \tilde{d}_z - (4.43 \times 10^{-1} - 0.57 \times 10^{-1}i)(y - \tilde{d}_y). \end{aligned}$$

However, we are more interested in what can be said about the triangular decomposition of the original non-linear problem.

First, we mention that the methods of [33] interpolate this one-dimensional curve as the intersection of $n + 1 = 4$ hyper-surfaces, which are in fact ruled surfaces. These ruled surfaces correspond to random (generic) projections. Lemma 5.3 in [33] expresses the fact that the irreducible component, corresponding to this one-dimensional curve, is precisely cut out by the intersection of these 4 ruled hyper-surfaces. Equivalently in terms of generic coordinates X, Y, Z, which are random linear combinations of x, y, z, the irreducible component is defined in terms of 4 polynomials in X, Y, Z. Three of the polynomials are bivariate polynomials (of (X, Y), (X, Z), and (Y, Z) respectively) and correspond to projections onto the generic coordinate hyperplanes. Triangular decomposition of this one-dimensional curve should require only 2 rather than 4 polynomials. However, approximate triangular representations in the generic coordinates can be extracted by choosing subsystems of 2 polynomials from the 3 bivariate polynomials. Note that, in general, such triangular representations will have excess components (removed by intersection with the remaining polynomials). In some cases, excess components do not occur (a fact that can be checked in a number of ways, e.g. by numerical membership testing, or testing numerically for irreducibility). In this case, a tight generic triangular representation is determined.

We now briefly discuss the task of finding triangular decompositions in terms of the original coordinates x, y, z, rather than the generic coordinates. It is clear from our comments above, in the linear case, that we need an ordering of the variables that is numerically stable, and should be applicable even in the non-linear case. Here, we require a projection that maps the nonlinear curve onto one which has the same dimension. The linearization shows that any projection will have this property. This projection can be interpolated, so the degree of the interpolated curve is important. The approach of [33] determines that the degree of the curve is 3, and this can only drop upon projection. So a second obvious property for a 'good' projection is that this degree not be diminished.

A possible alternative approach is to bootstrap from the zero-dimensional case. The general idea of reducing a positive dimensional system to a zero-dimensional one goes back to Van der Waerden [40] for primary decomposition, and was later made algorithmic by Gianni, Trager, and Zacharias [17], among others. The most recent work is due to Schost [31] and Dahan, et al. [11], where triangular decompositions are obtained for exact positive dimensional systems.

It is interesting to investigate how these methods might be adapted to approximate systems. Here is a sketch of this method. First, the numerical irreducible decomposition would be performed, and each irreducible component would be treated separately. The witness points for a component would be used to determine a stable ordering for the linearized system. Setting the free variables to random values yields a zero-dimensional system, to which the approximate equiprojectable decomposition [29] would be applied. The triangular sets so-obtained involve only the non-free variables. The dependence on the free variables could be determined, through choosing different random values for them, by interpolation. However, we caution that there are serious stability issues that need to be resolved, and also note that the interpolation process involving rational function reconstruction needs to be developed before such an approach can be realized.

Two-Dimensional Components: There is one single component of p, predicted by [33] which has dimension $n - 1 = 2$ and degree 2. Homotopy continuation can be used to generate enough points from which a single polynomial of degree 2 in x, y, z can be interpolated. This interpolation can be carried out automatically by PHCPack [41]. This is a triangular set for the component. Note also that the minimality of the degree ensures that the ideal is radical (one property of a triangular set). It is equivalent to a multiple of $x^2 + y^2 + z^2 - 1 = 0$, which is easily seen independently here, but, of course, the method will work on examples where such a property and an expression cannot be so easily extracted.

6. Discussion

Exact triangular decomposition methods for representing exact polynomials systems (see for example [43, 24, 28, 22, 42]) have proved valuable in applications [5, 23, 14, 31, 16], and there are well-developed algorithms [28, 9] for their construction. There have also been considerable recent improvements [12, 10] in the complexity of algorithms for their construction. Such representations are desirable, not only because of their triangular solved-form structure, but also because (in comparison with other exact methods) they give the minimum number of polynomials required to form a description of the equidimensional decomposition components of such systems.

The goal of our research is to extend such methods to approximate systems of polynomials. Our previous paper [29] gave a detailed treatment of zero-dimensional systems, and in this current paper we have studied linear positive dimensional systems using methods from Numerical Linear Algebra.

The methods of [33] enable approximate generic points on the solution components of polynomial systems to be computed by numerical homotopy continuation. These witness points give the layout of the decomposition as well as the number of witness points on a component, which is its degree. As we indicate in this paper, the zero-dimensional components, computed by the methods of [33] are a solved form (explicitly giving the coordinates of the zero sets), and constitute a collection of triangular sets (each being an isolated root). However, these sets do not directly correspond to the triangular representation computed by exact methods. Also, if n is the number of variables, each irreducible component of dimension $n-1$ (hyper-surfaces) can have additional generic points generated by homotopy continuation, and interpolated by a single polynomial. This representation, which is automatically generated by PHCPack of [41, 33], is again, as we note, trivially, a triangular set.

In the d-dimensional case where $0 < d < n-1$, the methods of [33] also generate additional points on each irreducible component, and give interpolating polynomials to represent the component. Each interpolating polynomial corresponds to a random projection of the component in $\mathbb{C}^n$ to a $(d+1)$-dimensional affine space. In general, $n+1$ generic projections (and $n+1$ interpolating polynomials) are used in that approach to precisely describe the component (see especially Lemma 5.3 of [33] for the theoretical justification). Such a representation is not triangular, since a triangular representation would only require $n-d$ polynomials. However, a triangular sub-system of $n-d$ polynomials can be extracted in generic coordinates (e.g. as discussed for the twisted cubic example). Although a numerical test for irreducibility can be applied, such generic triangular sub-systems are not generally irreducible. Their chance of irreducibility can be enhanced by adding random linear combinations of the remaining $d+1$ polynomials to the triangular system, while maintaining its triangular structure.

Given a point $\mathbf{x}^0$ on a d-dimensional irreducible component of a polynomial system computed using SVW, we compute a local linearization of the form $\mathbf{x} = \mathbf{x}^0 + P\boldsymbol{\alpha}$, where P is computed using the SVD, and the $\boldsymbol{\alpha}$'s are d newly introduced parameters. In the case of linear varieties this is actually a triangular representation. Secondly, by eliminating $\boldsymbol{\alpha}$, and provided certain stability-invertibility properties are satisfied, stable triangular representations in the variables $\mathbf{x}$ can be obtained for the linearization. We note that both of these representations involve $n-d$ linear polynomials. For linear components, [33] gives a non-triangular representation, also only requiring $n-d$ linear polynomials.

In future work, we will extend the results of this paper and [29] to the construction of triangular sets for positive dimensional systems of polynomials.

References

[1] D. N. Bernstein. *The Number of Roots of a System of Equations.* Functional Anal. Appl., 9(3): 183–185. Translated from Funktsional. Anal. i Prilozhen., 9(3): 1–4, 1975.

[2] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer-Verlag, New York, 1997.

[3] G. Chèze. *Absolute Polynomial Factorization in Two Variables and the Knapsack Problem.* J. Gutierrez, editor, Proc. ISSAC 2004, pages 87–94. ACM Press, 2004.

[4] G. Chèze and A. Galligo. *Four Lectures on Polynomial Absolute Factorization.* Solving Polynomial Equations: Foundations, Algorithms, and Applications, vol. 14 of Algorithms and Computation in Mathematics, pages 339–392. Springer-Verlag, 2005.

[5] S.-C. Chou. *Mechanical Geometry Theorem Proving.* D. Reidel Publ. Comp., Dordrecht, 1988.

[6] R. M. Corless, A. Galligo, I. S. Kotsireas, and S. M. Watt. *A Geometric-Numeric Algorithm for Factoring Multivariate Polynomials.* Proc. ISSAC 2002, pages 37–45. ACM Press, 2002.

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer-Verlag, New York, 2nd edition, 1997.

[8] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. *The Singular Value Decomposition for Polynomial Systems.* Proc. ISSAC '95, pages 96–103, ACM Press, 1995.

[9] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, and Y. Xie. *Equiprojectable Decompositions of Zero-Dimensional Varieties.* Proc. ICPSS, pages 69–71. University of Paris 6, France, 2004.

[10] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, and Y. Xie. *Lifting Techniques for Triangular Decompositions.* Proc. ISSAC 2005, pages 108–115. ACM Press, 2005.

[11] X. Dahan, X. Jin, M. Moreno Maza and É. Schost. *Change of Ordering for Regular Chains in Positive Dimension.* Accepted by Maple Conference, Canada, 2006.

[12] X. Dahan and É. Schost. *Sharp Estimates for Triangular Sets.* Proc. ISSAC 2004, pages 103–110. ACM Press, 2004.

[13] C. Eckart and G. Young. *The Approximation of one Matrix by Another of Lower Rank.* Psychometrika, vol. 1: 211–218, 1936.

[14] M. V. Foursov and M. Moreno Maza. *On Computer-Assisted Classification of Coupled Integrable Equations.* J. Symb. Comput., 33: 647–660, 2002.

[15] A. Galligo and D. Rupprecht. *Irreducible Decomposition of Curves.* J. Symb. Comput., 33(5): 661–677, 2002.

[16] X.-S. Gao and Y. Luo. *A Characteristic Set Algorithm for Difference Polynomial Systems.* Proc. ICPSS, pages 28–30. University of Paris 6, France, 2004.

[17] P. Gianni, B. Trager, and G. Zacharias. *Gröbner Bases and Primary Decomposition of Polynomial Ideals.* J. Symb. Comput., 6(2-3): 149–167, 1988.

[18] M. Giusti and J. Heintz. *La Détermination de la Dimension et des Points Isolés d'une Variété Algébrique Peuvent S'effectuer en Temps Polynomial.* D. Eisenbud and L. Robbiano, editors, Computational Algebraic Geometry and Commutative Algebra, Cortona 1991, volume XXXIV of Symposia Mathematica, pages 216–256. Cambridge University Press, 1993.

[19] M. Giusti and J. Heinz. *Kronecker's Smart, Little Black Boxes*, London Mathematical Society Lecture Note Series, vol. 284, pages 69–104. Cambridge University Press, 2001.

[20] M. Giusti, G. Lecerf, and B. Salvy. *A Gröbner Free Alternative for Polynomial System Solving.* J. Complexity, 17(1): 154–211, 2001.

[21] G. Golub and C. V. Loan. *Matrix Computations.* John Hopkins University Press, 3rd edition, 1996.

[22] M. Kalkbrener. *A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties.* J. Symb. Comput., 15: 143–167, 1993.

[23] I. A. Kogan and M. Moreno Maza. *Computation of Canonical Forms for Ternary Cubics.* Teo Mora, editor, Proc. ISSAC 2002, pages 151–160. ACM Press, 2002.

[24] D. Lazard. *Solving Zero-Dimensional Algebraic Systems.* J. Symb. Comput., 13: 117–133, 1992.

[25] G. Lecerf. *Computing the Equidimensional Decomposition of an Algebraic Closed Set by Means of Lifting Fibers.* J. Complexity, 19(4): 564–596, 2003.

[26] F. Lemaire, M. Moreno Maza, and Y. Xie. *The* `RegularChains` *Library.* Proc. Maple Conference 2005, pages 355–368, 2005. Distributed software with *Maple 10*, Maplesoft, Canada.

[27] A. Leykin and J. Verschelde. *Phcmaple: A Maple Interface to the Numerical Homotopy Algorithms in PHCpack.* Proc. ACA '04, pages 139–147, University of Texas at Beaumont, USA, 2004.

[28] M. Moreno Maza. *On Triangular Decompositions of Algebraic Varieties.* Technical Report 4/99, NAG, UK. Presented at the MEGA-2000 Conference, Bath, UK, 2000. http://www.csd.uwo.ca/∼moreno.

[29] M. Moreno Maza, G. Reid, R. Scott, and W. Wu. *On Approximate Triangular Decompositions I: Dimension Zero.* Proc. SNC 2005, pages 250–275, 2005.

[30] T. Sasaki. *Approximate Multivariate Polynomial Factorization Based on Zero-Sum Relations.* B. Mourrain, editor, Proc. ISSAC 2001, pages 284–291. ACM Press, 2001.

[31] É. Schost. *Complexity Results for Triangular Sets.* J. Symb. Comput., 36(3-4): 555–594, 2003.

[32] A. J. Sommese and J. Verschelde. *Numerical Homotopies to Compute Generic Points on Positive Dimensional Algebraic Sets.* J. Complexity, 16(3): 572–602, 2000.

[33] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Numerical Decomposition of the Solution Sets of Polynomial Systems into Irreducible Components.* SIAM J. Numer. Anal., 38(6): 2022–2046, 2001.

[34] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Using Monodromy to Decompose Solution Sets of Polynomial Systems into Irreducible Components.* C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher, editors, Application of Algebraic Geometry to Coding Theory, Physics and Computation, pages 297–315. Kluwer Academic Publishers, 2001.

[35] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Symmetric Functions Applied to Decomposing Solution Sets of Polynomial Systems.* SIAM J. Numer. Anal., 40(6): 2026–2046, 2002.

[36] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Numerical Irreducible Decomposition Using PHCpack.* Algebra, Geometry, and Software Systems, pages 109–130. Springer-Verlag, 2003.

[37] A. J. Sommese and C. W. Wampler. *Numerical Algebraic Geometry.* J. Renegar, M. Shub, and S. Smale, editors, Proc. AMS-SIAM Summer Seminar in Applied Mathematics, vol. 32 of Lectures in Applied Mathematics, 1995.

[38] H. J. Stetter. *The Nearest Polynomial with a Given Zero, and Similar Problems.* ACM SIGSAM Bulletin, 33(4): 2–4, 1999.

[39] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* Society for Industrial & Applied Mathematics, Philadelphia, 3rd edition, 1997.

[40] B. L. Van der Waerden. *Modern Algebra.* Frederick Ungar Publishing Co., New York, 2nd edition, 1953.

[41] J. Verschelde. *PHCpack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation.* ACM Transactions on Mathematical Software, 25(2): 251–276, 1999.

[42] D. Wang. *Elimination Methods.* Springer-Verlag, Wein, New York, 2001.

[43] W.-T. Wu. *On Zeros of Algebraic Equations — An Application of Ritt Principle.* Kexue Tongbao, 31(1): 1–5, 1986.

Marc Moreno Maza
ORCCA
Department of Computer Science
University of Western Ontario
London, Canada
e-mail: `moreno@orcca.on.ca`

Greg J. Reid, Robin Scott and Wenyuan Wu
e-mail: `reid@uwo.ca`
`rscott2@uwo.ca`
`wwu25@uwo.ca`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 299–313
© 2007 Birkhäuser Verlag Basel/Switzerland

On the Extended Iterative Proportional Scaling Algorithm

Ming-Deh Huang and Qing Luo

Abstract. The *iterative proportional scaling algorithm* is generalized to find real positive solutions to polynomial systems of the form: $\sum_{j=1}^{m} a_{sj}p_j = c_s$, $s = 1, \ldots, n$, where $p_j = \pi_j \prod_{s=1}^{n} x_s^{a_{sj}}$ with $a_{sj} \in \mathbb{R}$ and $\pi_j, c_s \in \mathbb{R}_{>0}$. These systems arise in the study of reversible self-assembly systems and reversible chemical reaction networks. Geometric properties of the systems are explored to extend the iterative proportional scaling algorithm. They are also applied to improve the convergent rate of the iterative proportional scaling algorithm when dealing with ill-conditioned systems. Reduction to convex optimization is discussed. Computational results are also presented.

1. Introduction

A real function of the form

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{K} \pi_i x_1^{\alpha_{1i}} x_2^{\alpha_{2i}} \cdots x_n^{\alpha_{ni}},$$

where $\pi_i > 0$ and $\alpha_{ji} \in \mathbb{R}$, is called a *posynomial*. We are interested in finding real positive solutions to posynomial systems of the form:

$$\sum_{j=1}^{m} a_{sj}p_j = c_s, \ s = 1, \ldots, n, \quad \text{where } p_j = \pi_j \prod_{s=1}^{n} x_s^{a_{sj}} \tag{1.1}$$

with $a_{sj} \in \mathbb{R}$, and $\pi_j, c_s \in \mathbb{R}_{>0}$. These systems arise in the study of algorithmic self-assembly. As will be explained in the next section, a solution to such a system corresponds to an equilibrium of a reversible self-assembly system [4] or a reversible chemical reaction network [6]. It will be shown that the real positive solution to such a system is unique when the underlying linear system:

$$\sum_{j=1}^{m} a_{sj}y_j = c_s, \ s = 1, \ldots, n,$$

has a real positive point.

A special case of the system (1.1) is where $a_{nj} = 1$ for all j; then letting $z = x_n$, the problem becomes one of finding a probability function of the form:

$$p_j = \pi'_j z \prod_{s=1}^{n-1} x_s^{a_{sj}},\ j = 1, \ldots, m, \tag{1.2}$$

that satisfies $\sum_j p_j = 1$ and $\sum_{j=1}^{m} a_{sj} p_j = c'_s$ for $s = 1, \ldots, n-1$, where $\pi'_j = \frac{\pi_j}{c_n}$ for $j = 1, \ldots, m$, and $c'_s = \frac{c_s}{c_n}$ for $s = 1, \ldots, n-1$. In this case the problem is exactly solving *maximum likelihood equations* in statistics (a good example is on page 114 of [7]). It is known that the positive solution is unique (see [3], also Chapter 4 of [2]). The solution can be found by a numerical algorithm called *iterative proportional scaling* [3]. With this method a simple transformation is applied to the system so that two additional conditions are satisfied: $\sum_{s=1}^{n} a_{sj} = 1$ and $\sum_{i=s}^{n} c_s = 1$. The following theorem proven in [3] can then be applied to solve for the solution.

Theorem 1.1. *Consider the system*

$$\sum_{j=1}^{m} a_{sj} p_j = c_s,\ s = 1, \ldots, n,$$

where

$$p_j = \pi_j \prod_{s=1}^{n} x_s^{a_{sj}},$$

$a_{sj} \geq 0$, $\sum_{s=1}^{n} a_{sj} = 1$,$c_s > 0$,*and* $\sum_{i=s}^{n} c_s = 1$. *Suppose that* $Ax = c$ *has a real positive solution, then the sequence* $\langle p^{(k)} : k = 0, 1, 2, \ldots \rangle$ *with* $p^{(k)} = (p_1^{(k)}, \ldots, p_m^{(k)})$ *and defined by* $p_i^{(0)} = \pi_i$, $p_i^{(n+1)} = p_i^{(n)} \prod_{r=1}^{s} \left(\frac{c_r}{c_r^{(n)}} \right)^{a_{ri}}$, *where* $c_r^{(n)} = \sum_{i=1}^{m} a_{ri} p_i^{(n)}$, *converges to the unique positive solution of the system.*

In this paper we generalize the iterative proportional scaling algorithm to solve the posynomial system (1.1). Our approach is to associate the system (1.1) with a parameterized family of systems S_v of the form 1.2 where $v \in \mathbb{R}_{>0}$. We define a function g on a suitable positive real interval such that for v in the interval, $g(v)$ is the value of z in the unique real positive solution to S_v. We show that the function $\frac{g(v)}{v}$ is decreasing in this interval and has a unique fixed point u. Moreover the solution to our system (1.1) can be easily obtained from that of S_u. The fact that g has a unique fixed point and $\frac{g(x)}{x}$ is decreasing allows us to devise a bisection strategy to find u by solving a sequence S_{v_i}, each using the iterative proportional scaling method. We show that if in the solution to our system, $\sum_i p_i = u^{-1}$ and the required precision is ϵ, then the number of times where we apply the iterative proportional scaling method can be bounded by $O(|\log u| + \log(1/\epsilon))$.

The rest of the paper is organized as follows. In Sect. 2 we discuss the geometric perspective of the system (1.1) and its application to self-assembly systems and reversible chemical reaction networks. In Sect. 3 we explore a special geometric property of the system (1.1). This property is applied in this section to improve the convergent rate of iterative proportional scaling for certain ill-conditioned cases.

It is also utilized in the next section in generalizing the proportional scaling algorithm. The details of the extended proportional scaling algorithm are presented in Sect. 4. We implement our algorithm in Mathematica 5.1. We discuss convex optimization as an alternative method for solving (1.1). Computational results are presented in Sect. 6, including comparison of the extended iterative proportional scaling method with convex optimization.

2. Geometric Perspective and Motivation from Self Assembly and Chemical Reaction Network

Given a set $\mathcal{B} = \{v_1, \ldots, v_d\}$ in $\mathbb{R}^m$ and a positive vector $\mu = (\mu_1, \ldots, \mu_d)$, let $V_{I_{\mathcal{B}},\mu}$ denote the set of of $x \in \mathbb{R}^m$ such that $x^{v_i^+} = \mu_i x^{v_i^-}$, for $i = 1, \ldots, d$, where v_i^+ and v_i^- are the nonnegative vectors with disjoint support such that $v_i = v_i^+ - v_i^-$. Suppose $v_1, \ldots, v_d \in \mathbb{Z}^m$. Then $V_{I_{\mathcal{B}},\mu}$ is the zero set of the ideal $I_{\mathcal{B},\mu} = \langle p^{v_i^+} - \mu_i p^{v_i^-} : v_i = v_i^+ - v_i^- \in \mathcal{B}\rangle$ in the polynomial ring $\mathbb{R}[p] = \mathbb{R}[p_1, \ldots, p_m]$. It is called the *deformed toric variety* of $\mathcal{B}$ under μ. When $\mu = (1, \ldots, 1)$, $V_{I_{\mathcal{B}},\mu}$ is simply called the *toric variety* $V_{I_{\mathcal{B}}}$.

For $a = (a_i) \in \mathbb{R}_{>0}$ and $b = (b_i) \in \mathbb{R}^m$, we define $a^b = \prod_i a_i^{b_i}$.

Let $A = (a_{sj})$ be an n by m real matrix and suppose $\mathcal{B} = \{v_1, \ldots, v_d\}$ spans the kernel of A. Let $\pi = (\pi_1, \ldots, \pi_m) \in \mathbb{R}^m_{>0}$, and $\mu_i = \pi^{v_i}$ for $i = 1,\ldots,d$. Then it can be shown that for $c = (c_1, \ldots, c_n)$ with $c_i > 0$ for all i, $p \in V_{I_{\mathcal{B}},\mu} \cap \{y \mid y > 0, Ay = c\}$ if and only if p yields a solution to (1.1). That is:

$$\sum_{j=1}^{m} a_{sj} p_j = c_s, \; s = 1, \ldots, n,$$

where

$$p_j = \pi_j \prod_{s=1}^{n} u_s^{a_{sj}}$$

with some $u \in \mathbb{R}^n_{>0}$.

Note that $V_{I_{\mathcal{B}},\mu} \cap \{y \mid y > 0, Ay = c\}$ is determined by the kernel of A, μ and c. Therefore we may assume without loss of generality that the matrix A is of rank n.

Let A_i be the i-th column of A for $i = 1, \ldots, m$. For $x \in \mathbb{R}^n_{>0}$, let $x^A = (x^{A_1}, \ldots, x^{A_m})$ and $\pi x^A = (\pi_1 x^{A_1}, \ldots, \pi_m x^{A_m})$. Then (1.1) can be rewritten as: $Ap = c$ where $p = \pi u^A$ with $u \in \mathbb{R}^n_{>0}$. Let $U = ker A$. Then $U^\perp = im(A^t)$. Suppose $c = Ap_0$ for some $p_0 \in \mathbb{R}^m_{>0}$. For $u \in \mathbb{R}^n_{>0}$, let $\mu = A^t \log u$, then $\mu \in im(A^t) = U^\perp$, and $p = \pi u^A = \pi e^\mu$ is a solution to (1.1) if and only if $Ap = c = Ap_0$ if and only if $p - p_0 \in U = ker A$. Hence our problem becomes one of finding $\mu \in U^\perp$ with $\pi e^\mu - p_0 \in U$, given positive vectors π and p_0. It follows from Proposition B.1 of [6] that such a μ is unique. Therefore, there is a unique positive solution to (1.1).

In a self-assembly system or a reversible chemical reaction network, we have a collection of species whose concentrations are represented by variables $x_1, \ldots,$

x_m respectively. Each complex is represented by a monomial x^a where $x = (x_i)$ and $a = (a_i)$ with $a_i \in \mathbb{N}$. Each reaction (or *event* in the terminology of [4]) is associated with a binomial $\sigma x^a - \tau x^b$ where σ is the forward rate and τ the backward rate. The rate of change of concentration of the j-th species with time is modeled by $\dot{x_j} = F_j(x)$ where $F_j(x) = -\sum_i(\sigma_i x^{a_i} - \tau_i x^{b_i})v_i(j)$ with $v_i = a_i - b_i$ and $v_i(j)$ is the j-th coordinate of the vector v_i. Let $F = (F_i)$, then the system $\dot{x} = F(x)$ models the dynamics of the mass-action system [4, 6].

Let $\mathcal{B} = \{v_1, \ldots, v_d\}$ in $\mathbb{R}^m$ and $\mu = (\mu_1, \ldots, \mu_d)$, with $\mu_i = \sigma_i/\tau_i$. Then for real positive vectors $p \in \mathbb{R}^m$, $F(p) = 0$ iff $p \in V_{I_{\mathcal{B},\mu}}$ [6]. Moreover if the initial condition is x_0 then the flow determined by the system of differential equations $\dot{x} = F(x)$ satisfies the condition that $x(t) - x_0$ is in the linear span of $\mathcal{B}$. Thus if we choose a matrix A such that $\ker(A) =< v_1, \ldots, v_d >$, then the positive intersection of the deformed toric variety $V_{I_{\mathcal{B},\mu}}$ and the set $\{x \mid x \geq 0, Ax = Ax_0\}$ is precisely the set of positive real equilibria of the flow defined by $\dot{x} = F(x)$ with initial condition x_0.

Example 1. Consider the following reversible chemical reactions:

1. $2H_2 + O_2 = 2H_2O$ with forward rate k_1 and backwards rate k_2,
2. $Cl_2 + H_2 = 2HCl$ with forward rate k_3 and backwards rate k_4.

If we represent the concentrations of O_2, Cl_2, H_2, H_2O and HCl by x_1, x_2, x_3, x_4 and x_5, then the binomial associated with the first reaction is $k_1x_3^2x_1 - k_2x_4^2$; the binomial associated with the second reaction is $k_3x_2x_3 - k_4x_5^2$. Let $x = (x_1, x_2, x_3, x_4, x_5)$ then the dynamical system associated with these two reactions is governed by the following differential equations:

$$\dot{x} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{pmatrix} = F(x) = \begin{pmatrix} k_2x_4^2x_2 - k_1x_3^2x_1x_2 \\ k_4x_5^2 - k_3x_2x_3 \\ -2(k_1x_3^2x_1x_2 - k_2x_4^2x_2) - (k_3x_2x_3 - k_4x_5^2) \\ 2(k_1x_3^2x_1x_2 - k_2x_4^2x_2) \\ 2(k_3x_2x_3 - k_4x_5^2)\,. \end{pmatrix}$$

Let $B = \{v_1 = (1, 0, 2, -2, 0), v_2 = (0, 1, 1, 0, -2)\}$ and $\mu = (k_2/k_1, k_4/k_3)$. Then $I_{\mathcal{B},\mu} =< x_1x_3 - \frac{k_2}{k_1}x_4^2, x_2x_3 - \frac{k_4}{k_3}x_5^2 >$ and all positive points in $V(I_{\mathcal{B},\mu})$ are equilibria.

Assume that the initial condition of the system $\dot{x} = F(x)$ is $\tilde{x}_0 = (\tilde{x}_1, \ldots, \tilde{x}_5)$ and we choose

$$A = \begin{pmatrix} 1 & 0 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 1 & 1/2 \end{pmatrix}$$

so that the kernel of A is spanned by B. Then the real positive equilibrium is $V(I_{\mathcal{B},\mu}) \bigcap \{x \mid x > 0, Ax = A\tilde{x}_0\}$.

To get the corresponding posynomial system, we need to find one positive vector $\pi = (\pi_1, \ldots, \pi_5)$ such that $\mu_i = \pi^{v_i}$; that is, $\pi_1\pi_3^2\pi_4^{-2} = \frac{k_2}{k_1}, \pi_2\pi_3\pi_5^{-2} = \frac{k_4}{k_3}$. Let $\pi_1 = \pi_2 = \pi_3 = 1, \pi_4 = \sqrt{\frac{k_1}{k_2}}, \pi_5 = \sqrt{\frac{k_3}{k_4}}$; then $\mu_i = \pi^{v_i}$, and the corresponding posynomial system is

$$\begin{pmatrix} 1 & 0 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 1 & 1/2 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = A\tilde{x}_0,$$

where $p_1 = u_1$, $p_2 = u_2$, $p_3 = u_3$, $p_4 = \sqrt{\frac{k_1}{k_2}} u_1^{\frac{1}{2}} u_3$, $p_5 = \sqrt{\frac{k_3}{k_4}} u_2^{\frac{1}{2}} u_3^{\frac{1}{2}}$.

3. Geometric Property

In this section, we show that the first $n-1$ equations in the algebraic system (1.1) can be considered as a curve parameterized by x_n. In other words, let $x_1 = g_1(x_n), \ldots, x_{n-1} = g_{n-1}(x_n)$ be the function derived from the first $n-1$ equation in (1.1). Then the function $f_n(g_1(x_n), \ldots, g_{n-1}(x_n), x_n)$ is increasing in $(0, +\infty)$. We denote the real positive domain as $\mathbb{P}$ in the following proposition.

Proposition 3.1. *Let $c = (c_1, \ldots, c_n) \in \mathbb{P}^n$ and $A = (a_{ij})$ be an n by m real matrix of rank n, and suppose that $Ax = c$ has a real positive solution. Let $\pi = (\pi_1, \ldots, \pi_m) \in \mathbb{P}^m$, and let*

$$f_i(x_1, \ldots, x_n) = \sum_{j=1}^{m} \pi_j a_{ij} x_1^{a_{1j}} \cdots x_n^{a_{nj}} - c_i, \; i = 1, \ldots, n.$$

Then the set of real positive points determined by

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0, \\ f_2(x_1, \ldots, x_n) = 0, \\ \quad \cdots\cdots \\ f_{n-1}(x_1, \ldots, x_n) = 0 \end{cases}$$

forms a curve $\{(X(x_n), x_n) \mid x_n \in \mathbb{R}_{>0}\}$ in $\mathbb{P}^n$ parametrized by x_n, where $X(x_n) = (x_1(x_n), \ldots, x_{n-1}(x_n))$ is such that $f_i(X(x_n), x_n) = 0$ for $i = 1, \ldots, n-1$. Moreover $f_n(X(a), a) \geq f_n(X(b), b)$ for any $a > b > 0$.

The proof of the proposition is in Appendix A.

One application of the proposition is in dealing with an ill-conditioned system where the exponent in one of the variables is unusually large or unusually small. In this case, the convergent rate of the iterative proportional scaling method tends to be slow. We can improve the convergent rate by finding the value of one variable using bisection while finding the values of the others using iterative proportional scaling method as follows.

Consider the system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & \cdots & a_{nm} \\ 1 & 1 \cdots & \cdots & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ 1 \end{pmatrix},$$

where

$$p_j = \pi_j \mu \prod_{s=1}^{n} x_s^{a_{sj}}.$$

Such a system can be translated into the system in Theorem 1.1 as discussed before, and thus can be solved by iterative proportional scaling method.

Without loss of generality, assume that a_{1n} is the exponent which is unusually large. From Proposition 3.1, we know that in the positive real domain the first $n-1$ equations in (1.1) determine a curve $\mathcal{L}$ parametrized by x_n. Thus for each real positive x_n there is a unique point $(g(x_n), x_n)$ on $\mathcal{L}$. Moreover f_n is an increasing function along $\mathcal{L}$, that is, $f_n(g(x_n), x_n)$ is increasing. If we can evaluate the function $g(x_n)$, then we can evaluate $f_n(g(x_n), x_n)$. Hence we can approximate the x_n-coordinate of the solution using the bisection method. To evaluate the function g at point $x_n = h$, we observe that after setting $x_n = h$, $g_1(h), \ldots, g_{n-1}(h)$ is the solution of the first $n-1$ equations form an algebraic system in $n-1$ variables of the same form as (1.1), with $\pi_i (1 \le i \le m)$ replaced by $\pi_i h^{a_{in}}$. Thus the reduced system has a unique real positive solution and can be solved by the iterative proportional scaling method. Moreover, the solution is none other than $g(h)$. If $f_n(g(h), h) > c_n$ and $(x_1^*, \ldots, x_n^*)$ is the solution of (1.1), then by Proposition 3.1, we know that $x_n^* < h$, otherwise $x_n^* \ge h$. We can repeat the above procedure to approximate x_n^* as close as we want. A precise description of the improved algorithm is given below:

1. Initially set $\mathit{startpoint} = 0, \mathit{endpoint} = c_n$.
2. Set $x_n = c_n/2$.
3. Let $\pi'_j = \pi_j * x_n^{a_{nj}}$. Solve the following system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n-1,1} & \cdots & \cdots & a_{n-1,m} \\ 1 & 1 \cdots & \cdots & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ 1 \end{pmatrix},$$

where

$$p_j = \pi'_j \mu \prod_{s=1}^{n-1} x_s^{a_{sj}},$$

by the iterative proportional scaling algorithm.

4. Compute $x_1, \ldots, x_{n-1}$ by solving the linear equation

$$\log p_j = \log \pi'_j + \log \mu + \sum_{s=1}^{n-1} a_{sj} \log x_s.$$

5. Compute $p_i = \pi_j \mu \prod_{s=1}^{n} x_s^{a_{sj}}$.
6. If $\sum a_{ni} p_i > c_n$, then let *endpoint* $= x_n$ and repeat from step 2. If $\sum a_{ni} p_i < c_n$, then let *startpoint* $= x_n$ and repeat from step 2. If $\sum a_{ni} p_i = c_n$, then return $(x_1, \ldots, x_n)$ as the solution.

4. The Algorithm

Consider a polynomial system of the form:

$$\sum_{j=1}^{m} a_{sj} p_j = c_s, \ s = 1, \ldots, n, \quad \text{where } p_j = \pi_j \prod_{s=1}^{n} x_s^{a_{sj}} \tag{4.1}$$

with $a_{sj} \in \mathbb{R}$, and $\pi_j, c_s \in \mathbb{R}_{>0}$. Let $\mu^{-1} = \sum_i p_i$, and $q_i = \mu p_i$. Then the system (4.1) is equivalent to

$$\sum_{j=1}^{m} a_{sj} q_j = \mu c_s, \ s = 1, \ldots, n; \quad \sum_i q_i = 1,$$

where

$$q_j = \pi_j \mu \prod_{s=1}^{n} x_s^{a_{sj}}.$$

For $v \in \mathbb{R}_{>0}$, let S_v be the system

$$\sum_{j=1}^{m} a_{sj} q_j = v c_s, \ s = 1, \ldots, n; \quad \sum_i q_i = 1, \tag{4.2}$$

where

$$q_j = \pi_j z \prod_{s=1}^{n} x_s^{a_{sj}}.$$

Note that S_v can be regarded as a posynomial system of the form (4.1) with $q_1, \ldots, q_m$ and z playing the role of $p_1, \ldots, p_m$. Therefore as discussed in Sect. 2, S_v has a unique positive solution if and only if (4.2) has a real positive solution. It is easy to verify that (4.2) has a positive solution if and only if $v \in [\alpha, \beta]$ where $\alpha = \min(\sum p_i)$ and $\beta = \max(\sum p_i)$ under the linear constraints $\sum_{j=1}^{m} a_{sj} p_j = c_s$, $s = 1, \ldots, n$; $p_i \geq 0$, $i = 1, \ldots, m$. Suppose $v \in [\alpha, \beta]$ and $x_1 = \rho_1$, $\ldots$, $x_n = \rho_n$, $z = \rho$ is the unique real positive solution to S_v; we define $\rho = g(v)$. We observe that if $v = g(v)$, then the system S_v is equivalent to (4.1). Moreover, the uniqueness of solution of (4.1) implies that the function g has a unique fixed point.

Proposition 4.1. *$g(x)/x$ is decreasing on $[\alpha, \beta]$.*

Proof. Suppose $v_1, v_2 \in [\alpha, \beta]$ and $v_1 < v_2$. Assume that $(x'_1, \ldots, x'_n, g(v_1))$ and $(x''_1, \ldots, x''_n, g(v_2))$ are the solution of S_{v_1} and S_{v_2} respectively. Then $(x'_1, \ldots, x'_n, z_1 = \frac{g(v_1)}{v_1})$ is the solution of

$$\sum_{j=1}^{m} a_{sj} q_j = c_s, \ s = 1, \ldots, n; \quad \sum_i q_i = 1/v_1,$$

where

$$q_j = \pi_j z \prod_{s=1}^{n} x_s^{a_{sj}},$$

and $(x''_1, \ldots, x''_n, z_2 = \frac{g(v_2)}{v_2})$ is the solution of

$$\sum_{j=1}^{m} a_{sj} q_j = c_s, \ s = 1, \ldots, n; \quad \sum_i q_i = 1/v_2,$$

where

$$q_j = \pi_j z \prod_{s=1}^{n} x_s^{a_{sj}}.$$

Now consider $x_1, \ldots, x_n$ as the function of z determined by $\sum_{j=1}^{m} a_{sj} q_j = c_s$, $s = 1, \ldots, n$; where $q_j = \pi_j z \prod_{s=1}^{n} x_s^{a_{sj}}$. From Proposition 3.1 in Sect. 3 we know that $\sum_i q_i$ is an increasing function in z. Let $\sigma(z) = \sum_i q_i(x_1(z), \ldots, x_n(z), z)$. Then $\sigma(z_i) = 1/v_i$ for $i = 1, 2$. Since $1/v_1 > 1/v_2$, we have $z_1 > z_2$. Hence $g(v_1)/v_1 \geq g(v_2)/v_2$ when $v_1 < v_2$. □

We have proved that $g(x)/x$ is decreasing on $[\alpha, \beta]$. We can use bisection search to find the fixed point of $g(x)$ on $[\alpha, \beta]$ (that is, the solution of $g(x)/x = 1$) since $g(x)/x$ is decreasing on $[\alpha, \beta]$.

Below we outline an algorithm for computing the fixed point u.

1. Compute the bound $[\alpha, \beta]$ for $1/\mu = \sum_i p_i$ under the linear constraints

$$\sum_{j=1}^{m} a_{sj} p_j = c_s, \ s = 1, \ldots, n; \quad p_j \geq 0, \ j = 1, \ldots, m,$$

by linear programming.

2. We start our bisection search for $1/\mu$ at $v_0 = \frac{\alpha+\beta}{2}$ (if $\beta = +\infty$, we can choose v_0 as arbitrary positive number bigger than α). Apply iterative proportional scaling algorithm to solve for:

$$\sum_{j=1}^{m} a_{sj} q_j = v_0 c_s, \ s = 1, \ldots, n; \quad \sum_i q_i = 1,$$

where

$$q_j = \pi_j z \prod_{s=1}^{n} x_s^{a_{sj}}.$$

3. Compute z through $\log z$ by solving the linear equation
$$\log q_j/\pi_j = \log z + \sum_{s=1}^{n} a_{sj} \log x_s, \ j = 1, \ldots, m.$$
4. If $z/v_0 = 1$, then obviously $q_j/v_0, j = 1, \ldots, m$, is the solution of (4.1). If $z/v_0 < 1$, then let $\beta = v_0$ and repeat step 2. If $z/v_0 > 1$, then let $\alpha = v_0$ and repeat step 2 (in this case, if $\beta = +\infty$, we set $v_0 = 2v_0$ when we repeat step 2). By doing this we can proceed to come close to $u = \sum_{i=1}^{n} p_i$ within a precision of ϵ using bisection search and compute $p_i = u * q_i$ is the solution of (4.1).

The convergence of iterative proportional scaling method has been proven in [3] and the convergence of bisection search has been proven in Proposition 4.1. Finally the real positive solution in (4.1) can be found by simple linear algebra once $p_i, i = 1, \ldots, m$, are known.

The running time of our algorithm is closely related with the convergent rate of iterative scaling method. In our algorithm, we apply $O(|\log u| + \log(1/\epsilon))$ times iterative proportional scaling method.

5. Reduction to Convex Optimization

As discussed earlier, our system can be interpreted as finding the intersection
$$V_{I_{\mathcal{B}},\mu} \cap \{y \mid y > 0, Ay = c\},$$
where the kernel of A is the linear span of B. A well-studied case is where we consider the intersection
$$V_{I_{\mathcal{B}}} \cap \{y \mid y > 0, Ay = c\}$$
with the additional assumption that $(1, \ldots, 1)$ is in the row space of A. In this case the unique solution is where the entropy function $-\sum_i p_i \log p_i$ is maximized in the convex set $\{y \mid y > 0, Ay = c\}$ (see p. 115 of [7]). In the more general situation, we do not assume that $(1, \ldots, 1)$ is in the row space of A. We can show that the unique solution to our system is where the function $-\sum_i (x_i \log \frac{x_i}{\pi_i} - x_i + \pi_i)$ is maximized over the convex set $\{y \mid y > 0, Ay = c\}$. The function is the relative entropy function adjusted by the difference in weights between x and π. It is precisely the negative of the *Kullback-Leibler divergence* function $D(x, \pi)$, which is known to be convex (see p. 90 of [1]). Hence the problem can be reduced to a convex optimization problem [1] of minimizing the convex function $D(x, \pi)$ over the convex set $\{x \mid x \geq 0, Ax = b\}$. The objective function $D(x, \pi)$ is not self-concordant (see p. 498 of [1]) and it is difficult to analyze the convergence rate when applying the general convex optimization method. Mathematica 5.1 has implemented the convex optimization as one of the built-in functions, but the error is much bigger than that of the extended iterative method when the entries in matrix A are relatively big. The comparison of computational results of these two algorithms is given in Sect. 6.3.

6. Computation

In this section we discuss some of our computational results.

6.1. Computation of Extended Iterative Proportional Scaling Algorithm

We implement our algorithm with Mathematica 5.1. The experiments show that when the the iterative proportional scaling algorithm runs well in the inner loop, the extended iterative proportional scaling algorithm works well in terms of accuracy and running time. The following is an instance of Example 1 in Sect. 2.

In this example,

$$A = \begin{pmatrix} 2 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 & 1 \end{pmatrix}.$$

Let $\pi = (1,1,1,1,1)$ and

$$c = \begin{pmatrix} 3.56081 \\ 10.0889 \\ 25.0121 \end{pmatrix}.$$

Then the solution of the above system is

$$(0.445231, 1.67631, 2.66594).$$

With 16625 iterations, the computation result from the extend iterative proportional scaling is

$$(0.445231, 1.67631, 2.66594),$$

which is exactly the actual solution of the system.

6.2. Improvement in the Iterative Proportional Scaling Method

We implement both the improved iterative proportional scaling method for dealing with ill-conditioned systems and the standard iterative proportional scaling method.

Example 2.

$$A = \begin{pmatrix} 40 & 1 & 3 & 3 & 2 \\ 1 & 4 & 3 & 2 & 5 \\ 0 & 1 & 4 & 3 & 1 \end{pmatrix},$$

$\pi = (1,2,3,4,5)$ and

$$c = \begin{pmatrix} 1.41519 \\ 4.38626 \\ 1.01694 \end{pmatrix}.$$

The value of p_i of the above system is

$$(0.30736 * 10^{-24}, 0.592453, 0.00165342, 0.00599196, 0.399902).$$

The standard iterative proportional scaling method takes 155868 iterations and get the value of p_i as

$$(0.30462 * 10^{-24}, 0.592453, 0.00165358, 0.00599177, 0.399902).$$

The improved iterative proportional scaling method takes 24444 iterations and get the value of p_i as

$$(0.3052 * 10^{-24}, 0.592453, 0.00165354, 0.005918, 0.399902).$$

In these computation results, we see that the output from both methods is close to the correct answer, but the iteration time of the improved iterative proportional method is much shorter.

6.3. Comparison

We compare the computation results of convex optimization method built in Mathematica 5.1 and that of the extended iterative proportional method. Since the optimization is already built in Mathematica 5.1, it will make sense to only compare the computation performance here. When the entries in the matrix A are small, both algorithms work well. When the entries in the matrix A get bigger, the performance of Convex Optimization tends to be unstable while the extended iterative proportional scaling method continues to perform well.

Example 3. Given

$$A = \begin{pmatrix} 6 & 1 & 0 & 6 & 0 \\ 2 & 6 & 3 & 4 & 2 \\ 5 & 3 & 1 & 2 & 1 \end{pmatrix} \text{ and } c = \begin{pmatrix} 1.4173575 \\ 6.876265 \\ 3.2714125 \end{pmatrix},$$

solve the posynomial system $Ap = c$, where $p_j = \prod_{s=1}^{3} x_s^{a_{sj}}$.

The convex optimization method in Mathematica 5.1 reports error for this question while the extended iterative proportional scaling method returns exactly the solution $x_1 = 0.75, x_2 = 1, x_3 = 0.8$, taking 11861 iterations.

Example 4. Given

$$A = \begin{pmatrix} 5 & 1 & 2 & 5 & 0 \\ 7 & 4 & 6 & 5 & 1 \\ 6 & 7 & 0 & 6 & 8 \end{pmatrix} \text{ and } c = \begin{pmatrix} 1.9043664000 \\ 4.9184137600 \\ 3.189678080 \end{pmatrix},$$

solve the posynomial system $Ap = c$, where $p_j = \prod_{s=1}^{3} x_s^{a_{sj}}$.

The convex optimization method in Mathematica 5.1 returns p_i as $p_1 = -5.46438 * 10^{-6}$, $p_2 = 0.356956$, $p_3 = 0.485795$, $p_4 = 4.35763 * 10^{-4}$, $p_5 = 0.115164$, which is obviously a wrong solution because p_1 is negative in the solution returned, while the extended iterative proportional scaling method returns exactly the solution $x_1 = 0.75, x_2 = 1, x_3 = 0.8$ with 60652 iterations.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[2] W. Fulton. *Introduction to Toric Varieties.* Princeton Unversity Press, 1993.

[3] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[4] L. Adleman. Toward a general theory of self-assembly. In *Foundation of Nanoscience Self-Assembled Architectures and Devices*, Snowbird, Utah, 2004.

[5] J. L. Troutman. *Variational Calculus with Elementary Convexity.* Springer-Verlag New York, 1983.

[6] M. Feinberg. The existence and uniqueness of steady states for a class of chemical reaction networks. *Arch. Rational Mech. Anal.*, 132:311–370, 1995.

[7] B. Sturmfels. *Solving Systems of Polynomial Equations.* Number 97. American Mathematical Society, 2002.

Appendix A.

Theorem A.1 ([5]). *Let A be an open set in R^{n+k} and let $f : A \to R^n$ be a C^r function. Write f in the form $f(x, y)$ where x and y are elements of R^k and R^n. Suppose that (a, b) is a point in A such that $f(a, b) = 0$ and the determinant of the $n \times n$ matrix whose elements are the derivatives of the n component functions of f with respect to the n variables, written as y, evaluated at (a, b), is not equal to zero; then there exists a neighborhood B of a in R^n and a unique C^r function $g : B \to R^k$ such that $g(a) = b$ and $f(x, g(x)) = 0$ for all $x \in B$.*

Proposition A.2. *For any $n \times m$ matrix E with rank n, if A is $m \times m$ positive definite matrix, then $A^{-1} - E^T(EAE^T)^{-1}E$ is nonnegative definite.*

Proof. Let $P = \begin{pmatrix} I_m & 0 \\ -(EAE^T)^{-1}E & I_n \end{pmatrix}$; then

$$P^T \begin{pmatrix} A^{-1} & E^T \\ E & EAE^T \end{pmatrix} P = \begin{pmatrix} A^{-1} - E^T(EAE^T)^{-1}E & 0 \\ 0 & EAE^T \end{pmatrix},$$

so $A^{-1} - E^T(EAE^T)^{-1}E$ is nonnegative definite if and only if

$$\begin{pmatrix} A^{-1} & E^T \\ E & EAE^T \end{pmatrix}$$

is nonnegative definite.

Notice that

$$\begin{pmatrix} A^{-1} & 0 \\ E & 0 \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & A^{-1} \end{pmatrix}\begin{pmatrix} A^{-1} & E^T \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} A^{-1} & E^T \\ E & EAE^T \end{pmatrix}.$$

For any vector v, since both A and A^{-1} are positive definite,

$$v\begin{pmatrix} A^{-1} & E^T \\ E & EAE^T \end{pmatrix}v^T = v\begin{pmatrix} A^{-1} & 0 \\ E & 0 \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & A^{-1} \end{pmatrix}\begin{pmatrix} A^{-1} & E^T \\ 0 & 0 \end{pmatrix}v^T \geq 0;$$

thus $A^{-1} - E^T(EAE^T)^{-1}E$ is nonnegative definite. □

Proposition A.3. *Let $c = (c_1, \ldots, c_n) \in \mathbb{P}^n$ and $A = (a_{ij})$ be an n by m real matrix of rank n, and suppose that $Ax = c$ has a real positive solution. Let $\pi = (\pi_1, \ldots, \pi_m) \in \mathbb{P}^m$, and let $f = (f_1, \ldots, f_{n-1})$ be a function mapping $\mathbb{R}^n \to \mathbb{R}^{n-1}$ defined by*

$$f_i = \sum_{j=1}^{m} \pi_j a_{ij} x_1^{a_{1j}} \cdots x_n^{a_{nj}} - c_i,\ i = 1, \ldots, n-1.$$

Then there exists a unique continuously differentiable function $g = (g_1, \ldots, g_{n-1}) : \mathbb{P} \to \mathbb{P}^{n-1}$ such that $f(g(t), t) = 0$ for all $t > 0$.

Proof. Let $C_{(n-1)\times(n-1)} = (c_{ij}) = (\frac{\partial f_i}{\partial x_j})$ for $0 < i, j < n$, and let

$$b_j = \pi_j x_n^{a_{nj}} \prod_{s=1}^{n-1} x_s^{a_{js}},\ 0 < j < m+1.$$

Then $c_{ij} = x_j^{-1} \sum_{k=1}^{m} a_{ik} a_{jk} b_k$ $(i \neq j)$ for $0 < i, j < n$.

Let

$$B = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1,n-1} & a_{2,n-1} & \cdots & a_{m,n-1} \end{pmatrix};$$

then $C = B\,\mathrm{diag}(b_1, \ldots, b_m)B^T$.

The rank of B is $n-1$ since the rank of A is n and it is easy to prove that C is a positive definite matrix when $(x_1, \ldots, x_n) \in \mathbb{P}^n$, and thus $\det(C) \neq 0$.

Suppose $v = (v_1, \ldots, v_n) \in P^n$ and $f(v) = 0$; then by Theorem A.1 in Appendix and the proof above, there exists a unique different function $g = g(x_n)$ defined in the neighborhood of v such that

$$f(g_1(x_n), \ldots, g_{n-1}(x_n), x_n) = 0.$$

Since for any given $t > 0$, there always exists a unique $u = (u_1, \ldots, u_{n-1})$ such that $f(u, t) = 0$, and from the uniqueness we must have $g(t) = (u_1, \ldots, u_{n-1})$. By Theorem A.1 in Appendix $g(t)$ is a continuously differentiable function in $(0, +\infty)$. □

Proposition A.4. *Let $F : R^{n+m} \to R^m$ be a C^r function. Write F in the form $F(x, y) = (F_1(x, y), \ldots, F_m(x, y))$ for $x \in R^n, y \in R^m$. Suppose $F(a, b) = 0$ and $\det(\frac{\partial(F_1,\ldots,F_m)}{\partial(y_1,\ldots,y_m)})(a, b) \neq 0$. Let $y(x) = (y_1(x), \ldots, y_m(x))$ be the implicit function defined in a neighborhood B of a so that $F(x, y(x)) = 0$ for all $x \in B$. Then for all $x \in B$,*

$$Dy(x) = -[D_y F(x, y)]^{-1} D_x F(x, y),$$

where

$$Dy(x) = \begin{pmatrix} (y_1)'_{x_1} & (y_1)'_{x_2} & \cdots & (y_1)'_{x_n} \\ (y_2)'_{x_1} & (y_2)'_{x_2} & \cdots & (y_2)'_{x_n} \\ \cdots & \cdots & \cdots & \cdots \\ (y_m)'_{x_1} & (y_m)'_{x_2} & \cdots & (y_m)'_{x_n} \end{pmatrix},$$

$$D_xF(x,y) = \begin{pmatrix} (F_1)'_{x_1} & (F_1)'_{x_2} & \cdots & (F_1)'_{x_n} \\ (F_2)'_{x_1} & (F_2)'_{x_2} & \cdots & (F_2)'_{x_n} \\ \cdots & \cdots & \cdots & \cdots \\ (F_m)'_{x_1} & (F_m)'_{x_2} & \cdots & (F_m)'_{x_n} \end{pmatrix},$$

$$D_yF(x,y) = \begin{pmatrix} (F_1)'_{y_1} & (F_1)'_{y_2} & \cdots & (F_1)'_{y_m} \\ (F_2)'_{y_1} & (F_2)'_{y_2} & \cdots & (F_2)'_{y_m} \\ \cdots & \cdots & \cdots & \cdots \\ (F_m)'_{y_1} & (F_m)'_{y_2} & \cdots & (F_m)'_{y_m} \end{pmatrix}.$$

Proof of Proposition 3.1. Let

$$b_j = \pi_j x_n^{a_{nj}} \prod_{s=1}^{n-1} x_s^{a_{js}},\ 0 < j < m+1,$$

$$C = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,m} \end{pmatrix},$$

$$U = \begin{pmatrix} b_1 & & \\ & \cdots & \\ & & b_m \end{pmatrix},\ X = \begin{pmatrix} x_1 & & \\ & \cdots & \\ & & x_{n-1} \end{pmatrix},\ v = \begin{pmatrix} a_{n1} \\ a_{n2} \\ \vdots \\ a_{nm} \end{pmatrix}.$$

Then

$$D_xF(x,y) = \begin{pmatrix} (f_1)'_{x_n} \\ (f_2)'_{x_n} \\ \vdots \\ (f_{n-1})'_{x_n} \end{pmatrix} = {x_n}^{-1} \begin{pmatrix} \sum_{j=1}^m a_{nj}a_{1j}b_j \\ \sum_{j=1}^m a_{nj}a_{2j}b_j \\ \vdots \\ \sum_{j=1}^m a_{nj}a_{n-1,n}b_j \end{pmatrix} = {x_n}^{-1}CUv,$$

$$D_y(F(x,y)) = \begin{pmatrix} (f_1)'_{x_1} & (f_1)'_{x_2} & \cdots & (f_1)'_{x_{n-1}} \\ (f_2)'_{x_1} & (f_2)'_{x_2} & \cdots & (f_2)'_{x_{n-1}} \\ \cdots & \cdots & \cdots & \cdots \\ (f_{n-1})'_{x_1} & (f_{n-1})'_{x_2} & \cdots & (f_{n-1})'_{x_{n-1}} \end{pmatrix} = CUC^TX^{-1}.$$

Note that

$$\frac{d\,f_n(X(x_n), X_n)}{dx_n} = \frac{d(\sum_{j=1}^m \pi_j a_{nj} x_1^{a_{1j}} \cdots x_n^{a_{nj}} - c_n)}{dx_n}$$

$$= \sum_{j=1}^m a_{nj}^2 \frac{b_j}{x_n} + \sum_{j=1}^m \sum_{i=1}^{n-1} b_j a_{nj} a_{ji} \frac{(x_i)'_{x_n}}{x_i} = x_n^{-1} v^T U v + v^T U^T C^T X^{-1} \begin{pmatrix} (x_1)'_{x_n} \\ \vdots \\ (x_{n-1})'_{x_n} \end{pmatrix}.$$

By Proposition A.4

$$\begin{pmatrix} (x_1)'_{x_n} \\ \vdots \\ (x_{n-1})'_{x_n} \end{pmatrix} = -D_y(F(x,y))^{-1}D_xF(x,y) = (-X(CUC^T)^{-1})(x_n^{-1}CUv).$$

Thus

$$\begin{aligned} \frac{d\,f_n(X(x_n),X_n)}{dx_n} &= x_n^{-1}v^TUv - x_n^{-1}v^TU^TC^T(CUC^T)^{-1}CUv \\ &= x_n^{-1}(v^TUv - v^TU^TC^T(CUC^T)^{-1}CUv) \\ &= x_n^{-1}((Uv)^T(U^{-1} - C^T(CUC^T)^{-1}C)Uv). \end{aligned}$$

By proposition A.2 in Appendix A, $U^{-1} - C^T(CUC^T)^{-1}C$ is nonnegative definite, $(Uv)^T(U^{-1} - C^T(CUC^T)^{-1}C)Uv \geq 0$, when $x_n > 0$, $\frac{d\,f_n(X(x_n),X_n)}{dx_n} \geq 0$. Hence $f_n(X(a),a) \geq f_n(X(b),b)$ for any $a > b > 0$. $\square$

Ming-Deh Huang and Qing Luo
Computer Science Department
University of Southern California
941 W. 37th Place, Los Angeles, CA 90089-0781 USA
e-mail: `huang@pollux.usc.edu`
`qingl@pollux.usc.edu`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 315–333
© 2007 Birkhäuser Verlag Basel/Switzerland

Preprocessing for Finite Element Discretizations of Geometric Problems

Hong Gu and Martin Burger

Abstract. In this paper, we use finite element methods to approximate the solutions of parameter-dependent geometric problems, and investigate the possibility of using symbolic methods as a preprocessing step. The main idea of our approach is to construct suitable finite element discretizations of the nonlinear elliptic equations leading to systems of algebraic equations, which can be subsequently solved by symbolic computation within the tolerance of computer algebra software. The prolongation of the preprocessed symbolic solution can serve as a starting value for a numerical iterative method on a finer grid.

A motivation for this approach is that usual numerical iterations (e.g. via Newton-type or fixed-point iterations) may diverge if no appropriate initial values are available. Moreover, such a purely numerical approach will not find all solutions of the discretized problem if there are more than one. A final motivation for the use of symbolic methods is the fact that all discrete solutions can be obtained as functions of unknown parameters.

In this paper, we focus on a special class of partial differential equations derived from geometric problems. A main challenge in this class is the fact that the polynomial structure of the nonlinearity is not explicit in the divergence form usually used for finite element discretization. As a consequence, the discrete form would always yield some non-polynomial terms. We therefore consider two different discretizations, namely a polynomial reformulation before discretization and a direct discretization of the divergence form with polynomial approximation of the discrete system. In order to perform a detailed analysis and convergence theory of the discretization methods we investigate some model problems related to mean-curvature type equations.

Keywords. Finite element method, symbolic computation, Newton iteration, preprocessing, multigrid, polynomial equation.

Supported by the Austrian Science Foundation Funds under grand SFB F013/F1304,F1308.

1. Introduction

Many geometric optimization problems can be formulated via a representation of the geometric unknown as a graph or implicitely in level set form. The corresponding optimality system then yields a system of nonlinear partial differential equations with special structure, like the famous Plateau problem (minimal surfaces) [9], free boundary problems in variational form (cf. e.g. [2]), problems in mathematical imaging (cf. e.g. [1]) or, similarly, in level set approaches to shape optimization and reconstruction (cf. e.g. [18, 4]). Similar problems also arise in implicit time discretizations of geometric evolution equations like the famous *mean-curvature flow*

$$\frac{\partial u}{\partial t} = \sqrt{p^2 + |\nabla u|^2}\ \mathrm{div}\ \left(\frac{\nabla u}{\sqrt{p^2 + |\nabla u|^2}} \right), \tag{1}$$

with $p = 1$ for a curve or surface represented as the graph of u, and $p = 0$ for the level set representation (cf. [18] for details). The appearance of purely geometric terms like the length of the surface element $Q := \sqrt{p^2 + |\nabla u|^2}$ or the mean curvature

$$H = \ \mathrm{div}\ \left(\frac{\nabla u}{\sqrt{1 + |\nabla u|^2}} \right)$$

is a typical characteristic of such geometric partial differential equation (by which we mean a differential equation derived from a geometric problem in graph or level set form). In many of these geometric equations real parameters (e.g. regularization parameters, cf. [4], or step sizes in time discretizations) appear, and it would be desirable to gain some information on the behavior of the solutions for different parameter values.

Due to the strong nonlinearity of differential operators involved in such geometric partial differential equation, it is often not straightforward to compute numerical solutions from a finite element discretization. For the solutions of the discrete method, Newton and fixed-point methods may fail to converge, in particular for parameter-dependent problems around critical parameter values. For discrete problems containing different roots (e.g. examples in Sect. 2), the numerical iterative methods are not sufficient to determine all roots and maybe may converge to undesired ones. On the other hand, if one can discretize the problem into a multivariate polynomial form, well-established symbolic methods for computing all the roots are available (cf. e.g. [21, 22]).

Since the discretized operator has similar approximates the continuous mapping, one may expect that the number of its roots might not depend strongly on the number of generated unknowns in the polynomial form, and at least it is limited by the BKK bound with respect to their sparse structure (cf. e.g. [6, 19]). The error of the polynomial coefficients caused by a proper numerical integration scheme will not significantly affect the number of roots either, according to the approximating theory associated to the finite element method(e.g. [5]) that provides stability. Following this argument one could convert all the floating point numbers appearing in the coefficients into rational numbers, for convenience of symbolic root computation. Practically, solving a large-scale parameter-dependent polynomial equation is more expensive than parameter-free cases, no matter whether we use symbolic computation or suitable numerical approaches. It is also important to construct a relatively cheap algorithm in order to extend the grid partition limit within the tolerance of computing software. For this sake we propose to use the symbolic method as a coarse grid solver within a multigrid framework. For the preprocessing step on the coarse grid, we do each symbolic elimination step [21] by the "subresultantChain" function from the computer algebra software "CASA" [15], that could be reasonably cheap for solving such sparse polynomial system derived by a finite element discretization. Based on the convergence theory related to this kind of multigrid approach [24], the finite element solution on the fine grid can strongly benefit from the coarse grid symbolic solution, if the range of parameters do not change the elliptic property of the linearization form. In previous related work (cf. e.g. [11]) such a multigrid method with symbolic preprocessing on the coarse grid has been considered already for the Plateau problem.

We shall investigate the numerical approach with preprocessed symbolic computation for a specific model problem of the form

$$QH = \sqrt{1+|\nabla u|^2}\,\operatorname{div}\left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right) = \varepsilon \qquad \text{in } \Omega, \tag{2}$$

subject to Dirichlet boundary conditions $u = f$ for a given sufficiently smooth function f on $\partial\Omega$, with Ω being a regular bounded domain in $\mathbb{R}^d$ ($d = 1$ or $d = 2$). Problem (2) is a geometric partial differential equation for the curve or surface determined by the graph of u, its geometric meaning is that length of a curve (or surface) element times the mean curvature are constant (a problem related to the prescribed curvature problem, cf. [8]). It seems clear for large curvatures (i.e. large ε) one cannot expect the curve or surface to be representable by a function graph, and consequently some singular behavior of (2) for large ε may appear, which makes the model problem challenging. Note that (2) is also close to a forward Euler time-discretization of the mean curvature flow of graphs (cf. [7])

$$\sqrt{1+|\nabla u|^2}\,\operatorname{div}\left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right) = \varepsilon(u-u_0), \tag{3}$$

where $\varepsilon = (\Delta t)^{-1}$ for the time step Δt and u_0 is the (given) initial value. In Sect. 2 we shall also consider the symbolic preprocessing applied to (3).

A major challenge in order to apply symbolic elimination methods to finite element discretization of nonlinear partial differential equations is to transfer the discrete nonlinear system into a system of algebraic equations. The obvious way of obtaining an algebraic system after discretization consists of rewriting the geometric partial differential equation as an equation with polynomial nonlinearity before discretization. This approach immediately yields algebraic equations after finite element discretization, but as we shall see below it may destroy a divergence structure in the equation so that the weak formulation involves also second derivatives. As a consequence, one might have to use very particular discrete subspaces and the convergence analysis of the discretization may require rather strong conditions. The convergence theory with respect to this kind of discretization should also be proved. We shall detail these issues in spatial dimension two.

As an alternative to the finite element discretization based on the algebraic form, one can also perform a discretization using a weak form related to the divergence structure, i.e.,

$$\int_\Omega \frac{\nabla u \cdot \nabla v + \varepsilon v}{\sqrt{1+|\nabla u|^2}}\ dx = 0, \tag{4}$$

for all suitable test functions v (following the classical approach by Johnson and Thomeé [16] for the minimal surface problem). Such a discretization does not yield an algebraic system, but as we shall show below, the discrete system can be approximated by an algebraic system after a simple perturbation. The perturbation is of higher order in terms of the discretization size and therefore will not destroy the convergence properties of the finite element method.

The paper is organized as follows: In the Sect. 2 we investigate the preprocessed symbolic solution of the model problem for one dimension ($d = 1$), i.e., the graph of u representing a curve, which serves to present the basic idea in a simple way. Section 3 is devoted to a discussion of the two-dimensional case, and the difficulties arising in the direct discretization of the algebraic form, for which we provide convergence results. In Sect. 4 we present results of computational experiments in the two-dimensional case and illustrate the properties of the parameter-dependent discrete solution(s) obtained by symbolic computation.

2. Preprocessed Computation of Curves

In this section, we initiate the idea by investigating (2) in the one-dimensional case $\Omega = [0,1]$, where

$$H = \ \mathrm{div}\ \left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right) = \frac{d^2u}{dx^2}\ \left(1+|\frac{du}{dx}|^2\right)^{-3/2}$$

represents the curvature of the curve $\Gamma = \{(x,u(x)) \mid x \in \Omega\}$, and $Q = \sqrt{1+|\frac{du}{dx}|^2}$ is the infinitesimal curve length. Using the relation for H, (2) can be rewritten as the elliptic differential equation

$$-\frac{d^2u}{dx^2} + \varepsilon(1+(\frac{du}{dx})^2) = 0 \quad \text{in} \ \ \Omega, \qquad u(0) = u(1) = 0. \tag{5}$$

For simplicity we restrict our attention to the case $f \equiv 0$, but analogous reasoning is possible for arbitrary boundary values.

The solution of (5) only exists for $-\pi < \varepsilon < \pi$ (for larger values of $|\varepsilon|$ the curves with prescribed curvature are not function graphs) and can be computed analytically in dependence of ε as

$$u(x;\varepsilon) = \frac{1}{\varepsilon}\left[\log\left(\cos\frac{\varepsilon}{2}\right) - \log\left(\cos(\varepsilon x - \frac{\varepsilon}{2})\right)\right].$$

The graph of u as a function of x and ε is illustrated in Fig. 1.

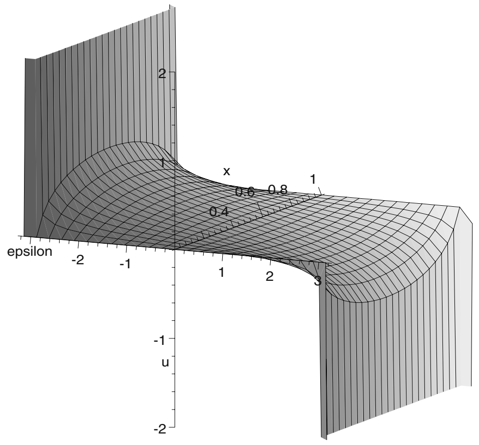


FIGURE 1. Plot of the exact solution $u(x;\varepsilon)$

2.1. Discretization of the Algebraic Form

Let $0 = x_0 < x_1 < \cdots < x_N < x_{N+1} = 1$ be a grid on the interval $[0,1]$ and let $h = \sup_i |x_{j+1} - x_j|$. We discretize the problem (5) into piecewise linear finite elements, i.e., we define the standard basis functions φ_j, $j = 1, 2, \ldots, N$, via

$$\varphi_j(x) = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}} & \text{if } x_{j-1} \le x \le x_j, \\ \frac{x_{j+1} - x}{x_{j+1} - x_j} & \text{if } x_j \le x \le x_{j+1}, \\ 0 & \text{else,} \end{cases}$$

and V^h =span$\{\varphi_j\}$. For $\varepsilon > 0$, we call $u_h(.;\varepsilon) \in V^h$ a discrete solution of (5) if

$$\int_0^1 \frac{du_h}{dx}(x;\varepsilon)\frac{dv}{dx}(x)dx + \varepsilon\int_0^1 (1 + \left(\frac{du_h}{dx}(x;\varepsilon)\right)^2)v(x)dx = 0,$$

for all $v \in V^h$.

If we use the unique representation in the form

$$u_h(x;\varepsilon) = \sum_{j=1}^{N} c_j(\varepsilon)\ \varphi_j(x),$$

with a vector $\mathbf{c}(\varepsilon) = (c_j(\varepsilon))_{j=1}^N \in \mathbb{R}^N$, then the finite element discretization yields a system of quadratic equations for the vector $\mathbf{c}(\varepsilon)$. The solution of this algebraic system is then computed directly in dependence of ε via symbolic elimination approaches.

For the parameter-dependent discrete solution, we computed on the grid $x_j = \frac{j}{N+1}$, $j = 1, ..N$ respectively. In all computational experiments we observed that indeed there exists no real solutions for ε large, in particular $|\varepsilon| > \pi$, and that the finite element solution is unique for $\varepsilon = 0$. For intermediate values of $|\varepsilon|$ however, we found two different real solutions of the discrete problem, with a lower branch approximating the real solution and an upper branch diverging as $\varepsilon \to 0$. The behavior is illustrated in Fig. 2 for $h = \frac{1}{4}$ and $h = \frac{1}{11}$, respectively, by plotting $u(x; \varepsilon)$ versus ε at $x = \frac{1}{2}$, respectively. One observes in particular that the second branch includes oscillations for the larger value of ε, which indicates its instability, but the first branch converge to the exact solution curve (the range of parameter ε also converge to $(-\pi, \pi)$ when the mesh size is refined).

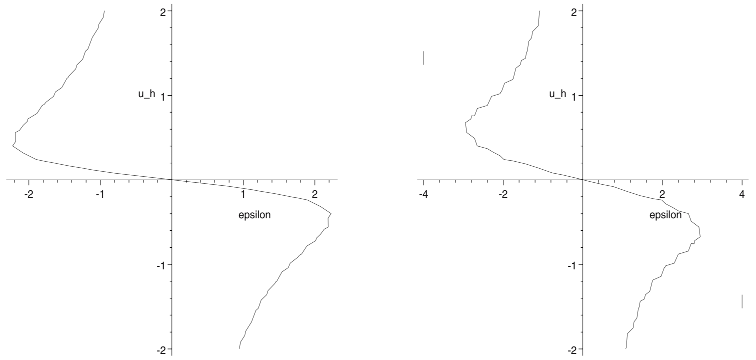


FIGURE 2. Finite element solution $u_h(\frac{1}{2}; \varepsilon)$ for $h = \frac{1}{4}$ (left) and $h = \frac{1}{11}$ (right)

From our numerical results, it becomes clear that a Newton-type method might converge to the wrong discrete solution. Even if additional continuation is used, further problems might appear for ε being close to π since the convergence radius will tend to zero. However, the direct symbolic computation does not meet such critical problem and it will be simpler to compute the result close to $|\varepsilon| = \pi$. The complexity of this symbolic approach increases exponentially with respect to the mesh refinement and the degree of polynomial equations. However, using the symmetry property

$$u_h(\frac{j}{N+1}) = u_h(\frac{N+1-j}{N+1}), \qquad j = 1, \ldots, N,$$

of the numerical solution one decreasing the size of the whole polynomial systems. Together with elimination taking into account by the sparsity of the polynomial form, the effort of the symbolic computation can be kept low. On a uniform grid with $h = 1/11$ (and thus 10 unknowns in the polynomial equations), the symbolic computation in the software package "CASA" only took few seconds.

Using an analogous approach, we can also compute discretize equation (3), which can be rewritten as

$$\varepsilon(u - u_0)(1 + \left(\frac{du}{dx}\right)^2) = \frac{d^2u}{dx^2}\,. \tag{6}$$

Due to the terms $u\left(\frac{du}{dx}\right)^2$ in the equation, the finite-element discretization generates a system of polynomials of degree three, and hence the complexity of the symbolic computation is higher than the previous case. However, one can still solve the coarse-grid system with reasonable effort. We illustrate the solution for $u_0 = x(1 - x)$ on $\Omega = [0, 1]$ by plotting $u_h(x = \frac{1}{2}, .)$ as a function of the parameter ε. One observes that the behaviour is rather complicated around $\varepsilon = 0$, but this would correspond to extremely large time steps in the mean-curvature flow. For the more interesting case of small time steps, there is a unique solution as one would expect from the original evolution problem.

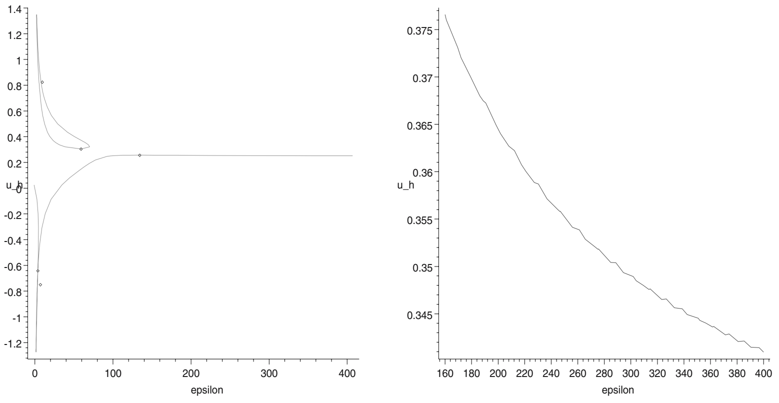


FIGURE 3. Finite element solution $u_h(\frac{1}{2}; \varepsilon)$ of (6) for $h = \frac{1}{7}$

2.2. Discretization of the Divergence Form

As an alternative of a direct discretization of the algebraic form, we consider an approach via the weak form (4). We use the same finite element discretization and basis functions as introduced in the previous section, for the sake of simplicity with $x_j = \frac{j}{N+1} = jh$ (but analogous reasoning is possible for arbitrary meshes). We consequently consider the computation of a discrete solution $\tilde{u}_h(., \varepsilon)$ satisfying

$$\int_\Omega \frac{\nabla \tilde{u}_h(., \varepsilon) \cdot \nabla v + \varepsilon v}{\sqrt{1 + |\nabla \tilde{u}_h(., \varepsilon)|^2}}\, dx = 0, \tag{7}$$

where in this case $\nabla = \frac{d}{dx}$. Using again the representation with respect to the basis functions in the form $\tilde{u}_h(x;\varepsilon) = \sum \tilde{c}_j(\varepsilon)\varphi_j(x)$, we obtain the discrete system (with $\tilde{c}_0 = \tilde{c}_{N+1} = 0$)

$$\frac{\tilde{c}_j - \tilde{c}_{j-1} + \frac{\varepsilon}{2}h^2}{\sqrt{h^2 + (\tilde{c}_j - \tilde{c}_{j-1})^2}} + \frac{\tilde{c}_j - \tilde{c}_{j+1} + \frac{\varepsilon}{2}h^2}{\sqrt{h^2 + (\tilde{c}_j - \tilde{c}_{j+1})^2}} = 0,$$

$j = 1, \dots, N$. We now multiply the equations by $h^{-2}Q_j^3$ where

$$Q_j = \sqrt{h^2 + \frac{1}{2}(\tilde{c}_j - \tilde{c}_{j-1})^2 + \frac{1}{2}(\tilde{c}_j - \tilde{c}_{j+1})^2},$$

to obtain the nonlinear system

$$A_j(\tilde{\mathbf{c}}) \quad = h^{-2}Q_j^3\left(\frac{\tilde{c}_j - \tilde{c}_{j-1} + \frac{\varepsilon}{2}h^2}{\sqrt{h^2+(\tilde{c}_j - \tilde{c}_{j-1})^2}} + \frac{\tilde{c}_j - \tilde{c}_{j+1} + \frac{\varepsilon}{2}h^2}{\sqrt{h^2+(\tilde{c}_j - \tilde{c}_{j+1})^2}}\right) = 0.$$

In order to obtain an approximating algebraic system, we use a Taylor expansion. More precisely, let $t_j := \frac{1}{2}(\tilde{c}_j - \tilde{c}_{j-1})^2 - \frac{1}{2}(\tilde{c}_j - \tilde{c}_{j+1})^2$; then

$$\frac{h^{-2}Q_j^3}{\sqrt{h^2+(\tilde{c}_j - \tilde{c}_{j-1})^2}} = \frac{h^{-2}Q_j^3}{\sqrt{Q_j^2+t_j}} = Q_j^2 - \tfrac{1}{2}h^{-2}t_j + O(t_j^2),$$
$$\frac{h^{-2}Q_j^3}{\sqrt{h^2+(\tilde{c}_j - \tilde{c}_{j+1})^2}} = \frac{h^{-2}Q_j^3}{\sqrt{Q_j^2-t_j}} = Q_j^2 + \tfrac{1}{2}h^{-2}t_j + O(t_j^2).$$

Under sufficient smoothness assumptions we have at least $t_j = O(h^2)$, and hence the perturbation is of order h^4. Ignoring higher-order terms with respect to t_j, we obtain the approximate equation operator

$$\begin{aligned} B_j(\mathbf{c}) &= (c_j - c_{j-1} + \tfrac{\varepsilon}{2}h^2)(1 + \tfrac{3}{4}h^{-2}(c_j - c_{j+1})^2 + \tfrac{1}{2}h^{-2}(c_j - c_{j-1})^2) \\ &\quad +(c_j - c_{j+1} + \tfrac{\varepsilon}{2}h^2)(1 + \tfrac{3}{4}h^{-2}(c_j - c_{j-1})^2 + \tfrac{1}{2}h^{-2}(c_j - c_{j+1})^2) \\ &= (2c_j - c_{j-1} - c_{j+1}) + \varepsilon(h^2 + \tfrac{1}{2}(c_j - c_{j+1})^2 + \tfrac{1}{2}(c_j - c_{j-1})^2) \\ &\quad +\tfrac{1}{4}h^{-2}(2c_j - c_{j-1} - c_{j+1})^3. \end{aligned}$$

We subsequently compute the discrete solution as

$$u_h(x;\varepsilon) = \sum c_j(\varepsilon)\varphi_j(x)$$

with $\mathbf{c}(\varepsilon)$ solving

$$B_j(\mathbf{c}(\varepsilon)) = 0, \qquad j = 1, \dots, N.$$

Note that the discrete operator $\tilde{B}$ corresponding to the discretization of the algebraic form in the previous section is given by

$$\tilde{B}_j(\mathbf{c}(\varepsilon)) = (2c_j - c_{j-1} - c_{j+1}) + \varepsilon(h^2 + \frac{1}{2}(c_j - c_{j+1})^2 + \frac{1}{2}(c_j - c_{j-1})^2)$$

so that the discrete equations only differ by the higher-order term $\frac{1}{4}h^{-2}(2c_j - c_{j-1} - c_{j+1})^3$. Consequently, we may expect similar behaviour of the discrete solutions, which is indeed confirmed by the numerical experiments. In particular, we obtain again two branches of solutions, as illustrated for $h = \frac{1}{4}$ and $h = \frac{1}{11}$ in Fig. 4. To compute these symbolic results is less expensive than applying the discrete method of Sect. 2.1, since the polynomial form are generated by low order expansion. But we could notice the convergence speed is almost the same as in Fig. 2.

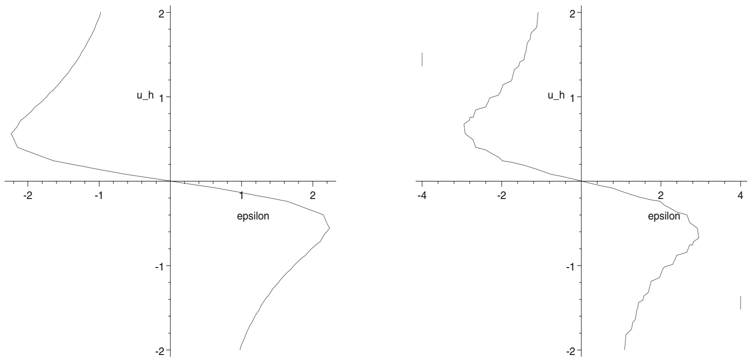


FIGURE 4. Finite solution $u_h(\frac{1}{2};\varepsilon)$ of (5) obtained from the divergence form, for $h=\frac{1}{4}$ (left) and $h=\frac{1}{11}$ (right)

3. Approximation of Surface Problems

We now consider problem (2) for $\Omega \subset \mathbb{R}^2$. With the notation $\frac{\partial u}{\partial x} = u_x$, $\frac{\partial^2 u}{\partial xy} = u_{xy}$, we can rewrite (2) in form as

$$\operatorname{div}\ ((1+u_y^2)u_x, (1+u_x^2)u_y) - 6u_{xy}u_xu_y + \varepsilon(1+u_x^2+u_y^2) = 0 \tag{8}$$

with the boundary value $u = f$ on $\partial\Omega$, where f is smooth function and Ω is a given regular domain.

Multiplying with a smooth test function v and integrating the divergence term by part, we obtain a weak formulation, namely to find u with $u = f$ on $\partial\Omega$ such that

$$\int_\Omega [(1+u_y^2)u_xv_x + (1+u_x^2)u_yv_y + 6u_{xy}u_xu_yv - \varepsilon(1+u_x^2+u_y^2)v] = 0, \tag{9}$$

for all sufficiently smooth functions v vanishing on $\partial\Omega$. Using a suitable finite element space $S_h(\Omega)$, the discretization consists in finding $u_h \in S_h(\Omega)$ with $u_h|_{\partial\Omega} = f_h$ satisfying

$$\begin{aligned}\int_\Omega &[(1+u_{h,y}^2)u_{h,x}v_x + (1+u_{h,x}^2)u_{h,y}v_y + 6u_{h,xy}u_{h,x}u_{h,y}v \\ &- \varepsilon(1+u_{h,x}^2+u_{h,y}^2)v] = 0,\ \forall\ v \in S_h^0(\Omega),\end{aligned} \tag{10}$$

where $S_h^0(\Omega)$ is the subspace of functions in $S_h(\Omega)$ vanishing on the boundary. The discretization then corresponds to an algebraic system for the coefficients with respect to a suitable set of basis functions, and this algebraic system is then solved by symbolic methods, which are introduced for the case $\varepsilon = 0$ in [10, 12].

A major difficulty in this approach is that the weak form still includes a mixed second derivative, which cannot be eliminated. In particular, in order to obtain a well-defined discrete form, the finite element space $S_h(\Omega)$ must admit mixed second derivatives, i.e.,

$$S_h(\Omega) \subset \{u \in W^{1,\infty}(\Omega) \mid u_{xy} \in L^2(\Omega)\}.$$

This property is satisfied e.g. for $S_h(\Omega)$ being a subspace of piecewise bilinear functions on a rectangular type mesh, where the edges are parallel to the coordinate axis, or for standard C^2 elements on arbitrary meshes and domains. In particular the finite element convergence analysis becomes rather complicated in such cases, as we shall work out in detail in the case of bilinear elements on a rectangular mesh.

The idea of using symbolic computation on a coarse grid has been used for the Plateau problem without parameter-dependence (cf. [11, 10]), and integrated into a two-grid algorithms with purely numerical computations on the fine grid (cf. [23, 24]). We shall discuss the solution by multigrid methods.

3.1. Convergence of the Discretization

To prove the convergence theory associated to the finite element approximation (10), we first need the following lemma:

Lemma 3.1. *Let $u \in W^{2,\infty}(\Omega)$ satisfy (8), and let for $v, w \in S_h(\Omega)$,*

$$R(u, w, v) := A(w, v) - A(u, v) - A'(u; w - u, v),$$

where

$$A'(u; w, v) = \int_\Omega (\sum_{i,j=1}^{2} a_{ij} w_{(i)} v_{(j)} + \sum_{i=1}^{2} b_i w_{(i)} v)$$

is the Jacobian of (8), with the coefficients $a_{ij}, b_i, c,\ i, j = 1, 2$ given by

$$a_{11}(u) = 1 + u_y^2,\ a_{22}(u) = 1 + u_x^2,\ a_{12}(u) = a_{21}(u) = -u_y u_x,$$

$$b_1(u) = 3u_y u_{xy} - 3u_{yy} u_x - 2\varepsilon u_x,$$

and

$$b_2(u) = 3u_x u_{xy} - 3u_{xx} u_y - 2\varepsilon u_y.$$

Then $u_h \in S_h(\Omega)$ solves (8) if and only if

$$A'(u; u_h, v) = R(u, u_h, v), \qquad \forall v \in S_h(\Omega).$$

Moreover, if

$$\|\partial_{xy} u_h\|_{0,\infty} + \|u_h\|_{1,\infty} \le K$$

and $S_h \Omega) = S_h^1 \Omega)$ is constructed by bilinear rectangular mesh interpolation, then the remainder R satisfies

$$R(u, u_h, v) \le C(K) \|u - u_h\|_{1,\infty}^2 \|v\|_{1,1}$$

with $C(K)$ independent of ε.

Proof. Set $G(t) = A(u + t(u_h - u), v)$. Then we have identity

$$G(1) = G(0) + G_t(0) + \int_0^1 G_{tt}(t)(1-t)dt.$$

By applying identity

$$\begin{aligned}
&\int_\Omega (w_{xy} - w_{h,xy})u_x u_y v \\
&= -\int_\Omega (w_x - w_{h,x})u_{xy}u_y v + (w_x - w_{h,x})u_x u_{yy} v + (w_x - w_{h,x})u_x u_y v_y \\
&= -\int_\Omega (w_y - w_{h,y})u_{xx}u_y v + (w_y - w_{h,y})u_x u_{xy} v + (w_y - w_{h,y})u_x u_y v_x,
\end{aligned}$$

then

$$\begin{aligned}
G_t(0) &= \partial_t|_{t=0} A(u + t(u_h - u), v) \\
&= \int_\Omega [(1 + u_y^2)(u_{h,x} - u_x)v_x + (1 + u_x^2)(u_{h,y} - u_y)v_y \\
&\quad +2u_x u_y(u_{h,x} - u_x)v_y + 2u_x u_y(u_{h,y} - u_y)v_x + 6u_{xy}u_y(u_{h,x} - u_x)v \\
&\quad +6u_{xy}u_x(u_{h,y} - u_y)v + 6u_x u_y(u_{h,xy} - u_{yx})v - 2\varepsilon u_x(u_{h,x} - u_x)v \\
&\quad -2\varepsilon u_y(u_{h,y} - u_y)v] \\
&= A'(u; w_h - w, v),
\end{aligned}$$

where the coefficients of A' satisfy the given conditions of the lemma.

And, by taking

$$R(u, u_h, v) := \int_0^1 G_{tt}(t)(1-t)dt,$$

a standard estimate using the Hölder inequality yields

$$\begin{aligned}
|R(u, u_h, v)| &\le \max|G_{tt}(t)| \\
&\le C(K)\|u - u_h\|_{1,\infty}^2 \|v\|_{1,1}.
\end{aligned}$$

Finally, if u_h solves (8), then $G(1) = 0$ and this completes the proof. □

We mention that the above estimate can be derived in an analogous way for any finite element discretizations $S_h(\Omega)$ if $\|u_h\|_{2,\infty} \le K$. The convergence theory can now be derived using the fact that $A'(u; \cdot, \cdot)$ is elliptic for $\varepsilon \neq 0$.

Theorem 3.1. *Let $\Omega = [0,1]^2$, $u \in W^{2,\infty}(\Omega)$ be a solution of (8) with $u = f$ on $\partial\Omega$. Moreover, let $S_h(\Omega)$ be a finite element subspace consisting of piecewise bilinear continuous functions on a rectangular grid. Moreover, let f_h be a linear interpolation f on the boundary segments of the grid. Then there exists a constant $c > 0$, such that for h sufficiently small, there exists a solution $u_h \in S_h(\Omega)$ of (10) with $u_h = f_h$ satisfying*

$$\|u_h - u\|_{1,\infty} \le c\, h. \tag{11}$$

Proof. For any solution u, we define a nonlinear operator $\Phi : S_h(\Omega) \to S_h(\Omega)$ via

$$A'(u; \Phi(v) - u, \phi) = R(u, v, \phi), \qquad \forall \phi \in S_h(\Omega). \tag{12}$$

Using ellipticity of A', it can be shown by standard arguments that Φ is well-defined and a continuous operator. Let P_h be the standard Galerkin projection operator associated to the bilinear form A', i.e., $P_h(u) = f_h$ on $\partial\Omega$ and

$$A'(u; P_h(u) - u, \phi) = 0, \ \forall \phi \in S_h^0(\Omega).$$

For $u \in W^{2,\infty}(\Omega)$, the assumptions of Theorem 8.1.11 and Corollary 8.1.12 in [3] are satisfied, which implies the existence of a positive constant C such that

$$\|u - P_h(u)\|_{1,\infty} \leq C\ h\ \|u\|_{2,\infty}.$$

Define the set

$$B = \{v \in S_h(\Omega) : \|v - P_h(u)\|_{1,\infty} \leq Ch\ \};$$

then by inverse estimates, there exist $C_1, C_2 > 0$ depending on u only, such that

$$\begin{aligned}
\|\partial_{xy}(v-u)\|_{0,\infty} &\leq \|\partial_{xy}(v - P_h(u))\|_{0,\infty} + \|\partial_{xy}(P_h(u) - u)\|_{0,\infty} \\
&\leq C_1 h^{-1} \|v - P_h(u)\|_{1,\infty} + C_2 \\
&\leq 2C_1 C + C_2.
\end{aligned}$$

Hence, $\|\partial_{xy} v\|_{0,\infty}$ is uniformly bounded with respect to h for v in the subspace $S_h(\Omega)$ of bilinear elements.

We now prove that $\Phi(B) \subset B$. In fact, when We substitute ϕ in (12) by using discrete Green functions $\phi = g_{h,x}^z$ and $\phi = g_{h,y}^z$, where

$$\begin{aligned}
A'(u; v, g_{h,x}^z) &= v_x(z), \\
A'(u; v, g_{h,y}^z) &= v_y(z),
\end{aligned}$$

apply the definition of Galerkin projection operator P_h, the last inequality of Lemma 3.1 and the properties of discrete Green functions (cf. [20]), then we obtain for all $v \in B$,

$$\begin{aligned}
\|\Phi(v) - P_h(u)\|_{1,\infty} &\leq C_0 |\log h| \|u - v\|_{1,\infty}^2 \\
&\leq 2C_0 |\log h| (\|P_h(u) - v\|_{1,\infty}^2 + \|u - P_h(u)\|_{1,\infty}^2) \\
&\leq 2C_0 |\log h| (C^2 h^2 + C^2 h^2) \\
&= 4C_0 C^2 |\log h| h^2 (\leq Ch),
\end{aligned}$$

for h sufficiently small.

By Brouwer's fixed point theorem, there exists a solution $u_h \in B$, such that $\Phi(u_h) = u_h$. And according to Lemma 3.1, u_h solves (10) and satisfies

$$\|u_h - u\|_{1,\infty} \leq \|u_h - P_h(u)\|_{1,\infty} + \|u - P_h(u)\|_{1,\infty} \leq 2Ch. \qquad \square$$

3.2. Discretization of the Divergence Form

In the following we discuss a two-dimensional version of the approximation of the divergence form (4), which can be performed on arbitrary triangular grids. For this sake we choose a standard finite element subspace V^h consisting of continuous piecewise linear functions on a triangular grid of size h. A finite element discretization consists in finding $\tilde{u}_h \in V_h$ satisfying

$$\int_\Omega \frac{\nabla \tilde{u}_h \cdot \nabla v + \varepsilon v}{\sqrt{1+|\nabla \tilde{u}_h|^2}}\, dx = 0, \qquad \forall\, v \in V_h. \tag{13}$$

With the standard set of nodal basis functions φ_j satisfying $\varphi_j(x_i) = \delta_{ij}$ for all grid points x_i we can represent the discrete solution as

$$\tilde{u}(x) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j(x).$$

Using the local support of the basis functions and the fact that $\nabla \tilde{u}_j$ is constant on each triangle, the finite element approximation (13) can equivalently be written as

$$Q_j^3 \sum_{T \in T(x_j)} \frac{1}{\sqrt{1+|\nabla \tilde{u}_h|_T|^2}} \int_T (\nabla \tilde{u}_h \cdot \nabla \varphi_j + \varepsilon \varphi_j)\, dx = 0,$$

$j = 1, \ldots, N$ with an arbitrary positive factor Q_j^3, where $T(x_i)$ is the set of triangles whose nodes include x_j. Now let $A_j = \sum_{T \in T(x_j)} |T|$ be the area of the triangles surrounding x_j, then we define

$$Q_j := \sqrt{1 + \sum_{T \in T(x_j)} \frac{|T|}{A_j} |\nabla \tilde{u}_h|_T|^2}.$$

Then we can derive a similar first-order Taylor expansion of

$$\frac{Q_j^3}{\sqrt{1+|\nabla \tilde{u}_h|_T|^2}} = \frac{Q_j^3}{\sqrt{Q_j^2 + t_j(T)}}$$

with respect to

$$\begin{aligned} t_j(T) &= |\nabla \tilde{u}_h|_T|^2 - \textstyle\sum_{T' \in T(x_j)} \frac{|T'|}{A_j} |\nabla \tilde{u}_h|_{T'}|^2 \\ &= \textstyle\sum_{T' \in T(x_j)} \frac{|T'|}{A_j} \left(|\nabla \tilde{u}_h|_T|^2 - |\nabla \tilde{u}_h|_{T'}|^2 \right). \end{aligned}$$

By similar reasoning as in the one-dimensional case we can derive an approximating weak form this way, which is polynomial (of order three) in the coefficients c_j. The arising system of algebraic system for the coefficients can subsequently be solved by symbolic elimination steps.

3.3. Multigrid Versions

At the current speed of symbolic elimination methods, the symbolic solution of the discretized problem can be carried out with reasonable efficiency by limited grid partition only. Therefore it seems natural to couple the symbolic solution technique with a multigrid approach (cf. [14] for an overview of multigrid methods), where symbolic solutions are computed on the coarse grid, and purely numerical techniques are used on finer grids.

Since the step between finer grids with numerical techniques is standard, we only discuss a two grid version, with a coarse grid (of size H) and a fine grid (of size h). In practical, the coarse finite element fomulation should be constructed reasonably within the tolerance of symbolic computing software. We assume to know suitable prolongation and interpolation operators $P_h^H : S_H(\Omega) \to S_h(\Omega)$ and $I_H^h : S_h(\Omega) \to S_H(\Omega)$, respectively. For given w, let $A'(w; \cdot, \cdot)$ be the bilinear map from Lemma 3.1. Then, we can immediately derive a cascadic multigrid method (similar to [11] for the parameter-free case) with exact symbolic solution on the coarse grid and a Newton-type correction at the fine grid which leads to higher accuracy [24, 10].

Cascadic Two-Grid Algorithm.

1. Compute $u_H \in S_H(\Omega)$ such that
$$A(u_H, v) = 0, \ \forall v \in S_H(\Omega).$$
2. Set $u_h^0 = P_h^H u_H$, and for $k = 1, \dots, k_*$ compute $u_h^k \in S_h(\Omega)$ from the linear equation
$$A'(u_h^{k-1}; u_h^k - u_h^{k-1}, v) = -A(u_h^{k-1}, v), \ \forall v \in S_h(\Omega).$$
3. Set $u_h = u_h^{k_*}$.

We finally mention that in an analogous way, standard V-cycle multigrid methods can be constructed, using the symbolic solver as a coarse grid correction, taking advantage of the fact that the exact coarse grid can also be computed in dependence of multiple parameters. Hence, if these parameters represent the values of the restriction of the fine grid solution, the symbolic coarse-grid solution can be computed in a preprocessing step, and during the V-cycle one only has to evaluate the coarse grid solution without extra computational effort.

4. Preprocessed Results for Surface Problems

In this section, we present the solutions of (10) obtained by the symbolic computation based on the computer algebra software "CASA" [15].

4.1. Discrete Solutions for Fixed Parameters

Set the boundary condition $f = 0$ and let the finite element space consists of piecewise-bilinear functions on a regular rectangular grid in the domain $\Omega =$

$[0, 1] \times [0, 1]$. We start with a grid consisting of 5×4 nodes, which produces (after elimination of the boundary nodes) 6 unknowns.

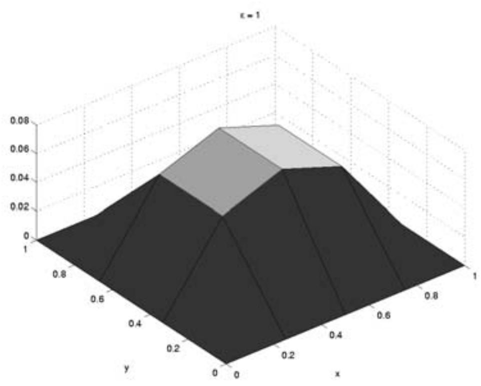


FIGURE 5. Finite element solution $u_h(\cdot; 1)$

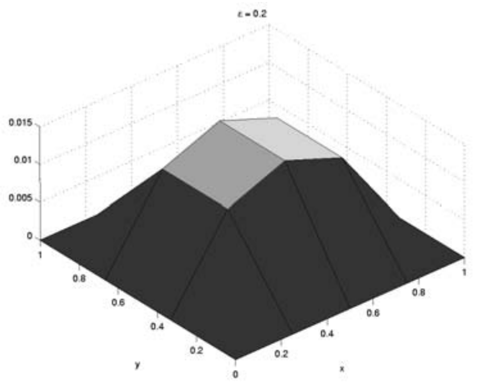


FIGURE 6. Finite element solution $u_h(\cdot; 1/5)$

We start by computing discrete solutions for fixed parameter ε on this mesh. The discrete solution turned out to be unique in all numerical experiments, which is a significant difference to the one-dimensional example, where the discrete solution was not unique or non-existent for a large range of the parameter values ε. We illustrate the results for $\varepsilon = 1$ and $\varepsilon = 1/5$ in Figs. 5 and 6.

If we change the average mesh to 5×5 nodes, we obtain a discretization fineness $h = 0.2$ and, after elimination of boundary nodes, 9 unknowns. The corresponding discrete solution for $\varepsilon = 1/5$ is plotted in Fig. 7.

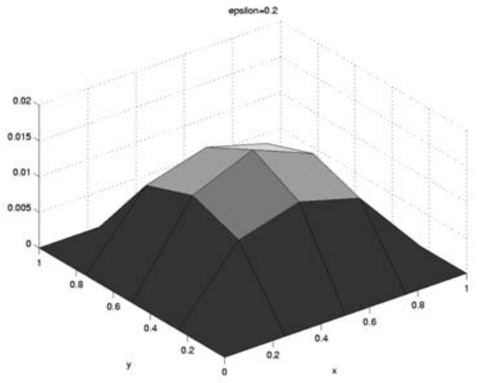


FIGURE 7. Finite element solution $u_h(\cdot; 1/5)$

4.2. Parameter-Dependent Discrete Solutions

However, we are actually solving the elliptic equations which contains parameters. The techniques do not make any difference if we use the symbolic eliminations on a parameter-dependent discrete system. In this case, one may consider u_h as a function of the parameter ε, a relation that can be illustrated by implicit function graphs at any point $(x, y) \in \Omega$. As an result, Fig. 8 shows the function graph of parameter-dependent solution $u_h(x = 0.5, y = 2/3; \varepsilon)$ based on a rectangular grid with 5×4 nodes on $\Omega = [0, 1]^2$.

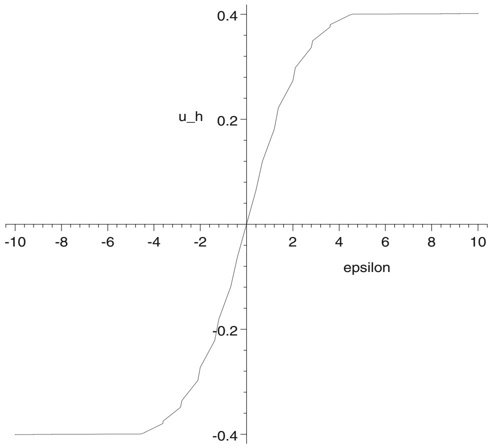


FIGURE 8. Function curve of $u_h(0.5, 2/3; \varepsilon)$

From an algebraic geometry point of view, the implicit function curve dependent on ε is isomorphic, and one observes that such elliptic problems in a two-dimensional domain contains a unique solution which does not depend on parameter ε. In other cases (such as in one dimension), where the existence or uniqueness of solutions might be changed with respect to the value of ε, hysteresis would be clearly reflected by singular points in those curves.

As a further remark of this section and Sect. 2, due to the complexity limit of using symbolic computing software, the accuracy of the obtained experimental results are limited by the size of the domain partition. However, these preprocessed results can be very close to the convergence radius associated to the Newton type methods according to the error estimates on the coarse grid. The preprocessing techiques can not only used to save the further Newton steps (which is not trival for the parameter-dependent case) but also help to determin the number of discrete solutions. The local accuracy can also be promoted efficiently by few numerical recorrection steps, like the multigrid approach or further iterative Newton step. The typical example shows the efficiency of approximating minimal surfaces by 2-grid agorithm can be seen in [11].

5. Conclusion and Further Remarks

This paper discusses a way of solving parameter dependent elliptic equations by finite element methods and preprocessing with symbolic computation, highlighting difficulties that are obtained when discretizing geometric partial differential equations into an algebraic form. A related method has also been considered for the regularization of certain ill-posed problems in [13] (where the parameter is a regularization parameter), and can be generalized to various classes of problems, whose nonlinearity can be rewritten into (or approximated by) a polynomial form.

The size of the coarse grid problem is still strongly limited, even if the symbolic approach took the advantage of the sparsity structure in the discrete problem. However, the preprocessing helps to obtain numerical solutions on fine grids with lower effort and higher reliability.

Acknowledgements

This work has been partially supported by the Austrian National Science Foundation (FWF) through the grant SFB F013, subprojects 04 and 08. The authors thank Prof. Franz Winkler and Prof. Ulrich Langer (University of Linz) for stimulating discussions and suggestions.

References

[1] L. Alvarez, F. Guichard, P. L. Lions and J. M. Morel, *Axioms and fundamental equations of image processing*, Arch. Rat. Mech. Anal. **123** (1993), 199–257.

[2] E. Bonnetier, R. Falk and M. Grinfeld, *Analysis of a one-dimensional variational model of the equilibrium shape of a deformable crystal*, Math. Model. Anal. Numer. **33** (1999), 573–591.

[3] S. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, 1994.

[4] M. Burger and S. Osher, *A survey on level set methods for inverse problems and optimal design*, Preprint, 2004.

[5] P. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, 1978.

[6] D. Cox, J. Little and D. O'Shea, *Using Algebraic Geometry*, Springer, New York, 1998.

[7] K. Deckelnick and G. Dziuk, *Error estimates for a semi-implicit fully discrete finite element scheme for the mean curvature flow of graphs*, Interfaces Free Bound. **2** (2000), 341–359.

[8] G. Dziuk and J. E. Hutchinson, *Finite element approximations and the Dirichlet problem for surfaces of prescribed mean curvature*, in Mathematical Visualization: Algorithms, Applications, and Numerics (H. C. Hege et al., eds.), Springer, Berlin, 1998, 73–87.

[9] J. Gray, *The Hilbert Challenge*, Oxford University Press, 2000.

[10] H. Gu, *Numerical Methods and Symbolic Computations for Generating Minimal Surfaces*, PhD thesis, RISC-Linz, April 2003.

[11] H. Gu, *Graphical generating of minimal surfaces subject to the Plateau problems*, SFB-Report 02-24, SFB F013, Johannes Kepler University, 2002.

[12] H. Gu, *Generating minimal surfaces subject to the Plateau problems by finite element method*, Numerical Methods and Applications, 5th International Conference NMA '02, Borovets, Bulgaria, Springer Lecture Notes in Computer Science 2542, 471–478, 2002.

[13] H. Gu and S. Kindermann, *Solution analysis of PDEs related to the Mumford-Shah functional with symbolic computation*, SFB-Report 03-17, SFB F013, Johannes Kepler University, 2003.

[14] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer, Berlin, 1985.

[15] R. Hemmecke, E. Hillgarter, and F. Winkler *CASA*, in Handbook of Computer Algebra: Foundations, Applications, Systems (J. Grabmeier, E. Kaltofen and V. Weispfenning, eds.), Springer-Verlag, 2003, 356–359.

[16] C. Johnson and V. Thomeé, *Error estimates for a finite element approximation of a minimal surface*, Math. Comp. **29**/130 (1975), 343–349.

[17] Q. Lin and Q. Zhu, *The Preprocessing and Postprocessing for the Finite Element Method*, Shanghai Sci. and Tech. Publishers, 1994 (in Chinese).

[18] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, 2002.

[19] F. Sottile, *Enumerative real algebraic geometry*, in Algorithmic and Quantitative Real Algebraic Geometry, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 60, Amer. Math. Soc., Providence, RI, 2003, 139–179.

[20] V. Thomeé, J. Xu and N. Zhang, *Superconvergence of gradient in piecewise linear finite element approximation to a parabolic problem*, SIAM J. Numer. Anal. **26** (1989), 389–414.

[21] D. Wang, *Elimination Methods*, Springer-Verlag, Wien, New York, 2001.

[22] F. Winkler, *Polynomial Algorithms in Computer Algebra*, Springer-Verlag, Wien, New York, 1996.

[23] J. Xu, *Two-Grid discretization techniques for linear and nonlinear partial differential equation*, SIAM J. Numer. Anal. **33** (1996), 1759–1777.

[24] J. Xu and A. Zhou, *Local and parallel finite element algorithms based on two-grid discretizations for nonlinear problems*, Adv. Comput. Math. **14** (2001), 393–327.

Hong Gu
Research Institute for Symbolic Computation
Johannes Kepler University
4232, Hagenberg, Austria
e-mail: `hgu@risc.uni-linz.ac.at`

Martin Burger
Industrial Mathematics Institute
Johannes Kepler University
4040, Linz, Austria
e-mail: `martin.burger@jku.at`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 335–347
© 2007 Birkhäuser Verlag Basel/Switzerland

Symbolic Computation Sequences and Numerical Analytic Geometry Applied to Multibody Dynamical Systems

Wenqin Zhou, David J. Jeffrey and Greg J. Reid

Abstract. The symbolic-numeric computing described here consists of an extensive symbolic pre-processing of systems of differential-algebraic equations (DAE), followed by the numerical integration of the system obtained. The application area is multibody dynamics. We deal symbolically with a DAE system using differentiation and elimination methods to find all the hidden constraints, and produce a system that is leading linear (linear in its leading derivatives). Then we use LU symbolic decomposition with Large Expression Management to solve this leading linear system for its leading derivatives, thereby obtaining an explicit ODE system written in terms of computation sequences obtained from using the Maple package `LargeExpressions`. Subsequently the Maple command `dsolve` is applied to this explicit ODE to obtain its numeric solution. Advantages of this strategy in avoiding expression explosion are illustrated and discussed. We briefly discuss a new class of methods involving Numerical Algebraic and Analytic Geometry.

Mathematics Subject Classification (2000). Primary 70E55; Secondary 68U01; Tertiary 15A09.

Keywords. LU symbolic decomposition, large expression management, differential elimination, DAE, computation sequence, straight line programme, hidden constraint, computer algebra.

1. Introduction

In the study of multibody dynamics, both purely symbolic and purely numeric methods separately suffer drawbacks. The symbolic methods encounter large expressions and the numerical methods are very slow in formulating the system. Examples of the applications of multibody dynamics are robot arms, vehicle suspensions, and automatic barriers. A modern robot arm, such as the Space Shuttle's Remote Manipulator System (RMS, or Canadarm) consists of several rigid bodies (the arm segments and the end effectors) linked by joints. Therefore the systems have at least 7 degrees of freedom, and typically more. In order to simulate the behaviour of systems like these, a number of programs have been developed that automatically generate the equations of motion for the system from its engineering description [3]. Several computer-algebra based packages have been developed, for example DYNAFLEX and SYMOFROS, that will generate the equations of motion in symbolic form. At present, these equations are handed over to purely numerical systems for integration as soon as the symbolic system has generated the equations.

What is described in this paper is a closer coupling of the symbolic analysis of the mechanical system and the final numerical integrations used by a simulation. The coupling takes the form of a symbolic pre-processing of the equations that improves the efficiency of their numerical solution. This increased interaction between the symbolically based part of the computation and the numerical part deserves to be called a "symbolic–numeric" calculation, we feel, even though the calculation is quite different from the numerical polynomial algebra [4] that has so far been the subject covered by the description "symbolic–numeric computation".

The equations describing the dynamics of a multibody system take the general form

$$\begin{aligned} M(t,q,\dot{q})\ddot{q} + \Phi_q^T \lambda &= F(t,q,\dot{q}) \ , \qquad (1) \\ \Phi(t,q) &= 0 \ . \qquad (2) \end{aligned}$$

Here, q is a vector of generalized co-ordinates, $M(t,q,\dot{q})$ is the mass matrix, Φ is a vector of the constraint equations and λ is a vector of Lagrange multipliers [1, 2]:

$$\Phi = \begin{bmatrix} \Phi^1 \\ \vdots \\ \Phi^m \end{bmatrix} , \qquad \Phi_q = \begin{bmatrix} \Phi^1_{q_1} & \cdots & \Phi^1_{q_n} \\ \vdots & \vdots & \vdots \\ \Phi^m_{q_1} & \cdots & \Phi^m_{q_n} \end{bmatrix} , \qquad \lambda = \begin{bmatrix} \lambda^1 \\ \vdots \\ \lambda^m \end{bmatrix} .$$

The Lagrange multipliers λ can be interpreted as the forces that the joints must exert between the elements in order to enforce the geometrical constraints they impose.

These symbolic models are too complicated to be solved symbolically, both because of their nonlinearity and because of the number of equations. However there is still the possibility of symbolically pre-processing them before attempting a numerical solution. It can be noted from the general form of the equation system (1)–(2) that it is differential-algebraic (DAE), with the numerical difficulties that that entails. Suitable objectives for the pre-processing include the simplification of the system and the determination of all constraints in order to facilitate the subsequent numerical simulation of the system. More specifically, we show below that we can convert the system to a purely differential one, and moreover separate the constraint forces λ from the simulation variables q. Our primary tool for this is the RIFSIMP package, which uses differentiation and elimination methods to simplify any over-determined polynomially nonlinear PDE or ODE system and to return a canonical differential form [7].

Direct application of the RifSimp package to multibody systems reveals that it has difficulty handling the large systems generated by Dynaflex. We use Implicit Reduced Involutive Form (IRIF) to assist in alleviating such large expression swell problems [10]. Implicit RIF form is converted to RIF form by symbolically solving the IRIF for its leading linear derivatives. In particular we use symbolic LU decomposition with large expression management to obtain RIF form, expressed in terms of computation sequences.

In this paper, we use the simple example of a two-dimensional slider crank to illustrate our approach to the symbolic-numeric solving of this kind of DAE system. After receiving the DAE model of the slider crank from Dynaflex we first use implicit RifSimp to compute all the hidden constraints of this higher index DAE, by reducing it to an index one or zero DAE system. The details are given in Sects. 2 and 3. Then in Sect. 4, we use symbolic LU matrix factoring with large expression management to get the canonical form for the differential equations, which helps the numerical integration. Using these symbolic equations with computation sequences, we illustrate the numerical simulation in Sect. 5. Finally in Sect. 6, we discuss some new ideas for solving the constraints to obtain consistent initial values for the integration. It should be emphasized that each of the sub-tasks described here are fully algorithmic and automated computations. Their integration into a single piece of software is in principle straightforward.

2. Two-Dimensional Slider Crank

Space limitations prevent us from describing a realistic mechanical system, both because the system description would take up significant space, and also because the equations generated by Dynaflex would by too lengthy for the reader to follow. The two-dimensional slider crank is a simple example of a closed-loop system with $q^T = (\theta_1, \theta_2)^T$ where $\theta_1 = \theta_1(t)$ and $\theta_2 = \theta_2(t)$ are the angles shown in Fig. 1. The system is given by (1)–(2) where:

$$M = \begin{bmatrix} l_1^2(\frac{1}{4}m_1 + m_2 + m_3) + J_1 & -l_1 l_2 \cos(\theta_1 + \theta_2)(\frac{1}{2}m_2 + m_3) \\ -l_1 l_2 \cos(\theta_1 + \theta_2)(\frac{1}{2}m_2 + m_3) & l_2^2(\frac{1}{4}m_2 + m_3) + J_2 \end{bmatrix}, \quad (3)$$

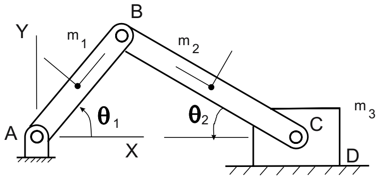


Figure 1. The two-dimensional slider crank. The arm of length l_1 and mass m_1 rotates while the mass m_3 attached to the end of the arm of length l_2 moves left and right. Each arm has mass m_i and moment of inertia J_i, for $i = 1, 2$.

$$F = \begin{bmatrix} -l_1 g(\frac{1}{2}m_1 + m_2 + m_3)\cos\theta_1 - l_1 l_2 \dot{\theta_2}^2 \sin(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) \\ l_2 g(\frac{1}{2}m_2 + m_3)\cos\theta_2 - l_1 l_2 \dot{\theta_1}^2 \sin(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) \end{bmatrix}, \tag{4}$$

and there is a single constraint equation between the angles:

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0 \ . \tag{5}$$

Therefore, $\Phi_q^T\lambda$ in (1) is given by $\Phi_q^T\lambda = \begin{pmatrix} l_1\cos\theta_1 \\ -l_2\cos\theta_2 \end{pmatrix}\lambda$. Note that in this example, λ is a scalar. Thus, in addition to generating the constraint (5), DYNAFLEX automatically generated the constraint force $\lambda(t)$. For this example, the challenge now is to analyze the equations with computer algebraic methods such as RIFSIMP.

3. Implicit Reduced Involutive Form

We use implicit RIFSIMP to get all hidden constraints in the multibody dynamical general model (1)–(2).

Definition [Implicit Reduced Involutive Form]. Let $\prec$ be a ranking. Then a system $L = 0, N = 0$ is said to be in implicit reduced involutive form if there exist derivatives $r_1, \ldots, r_k$ such that L is leading linear in $r_1, \ldots, r_k$ with respect to $\prec$ (i.e. $L = A[r_1, \ldots, r_k]^T - b = 0$) and

$$[r_1, \ldots, r_k]^T = A^{-1}b \ , \quad N = 0 \ , \quad \det(A) \neq 0 \tag{6}$$

is in reduced involutive form.

This form is of interest because computing A^{-1} symbolically in practice can be very expensive. Sometimes implicit RIF-form can be obtained very cheaply, just by appropriate differentiation of the constraints.

To convert a system of general form (1), (2) with non-trivial constraints to implicit reduced involutive form, one would have to at least differentiate the constraints twice [10]. Carrying this out we obtain

$$\begin{aligned} M\ddot{q} + \Phi_q^T\lambda &= F(t,q,\dot{q}), & (7)\\ D_t^2\Phi = \Phi_q\ddot{q} + H\dot{q} + 2\Phi_{tq}\dot{q} + \Phi_{tt} &= 0, & (8)\\ D_t\Phi = \Phi_q\dot{q} + \Phi_t &= 0, & (9)\\ \Phi(t,q) &= 0, & (10) \end{aligned}$$

where $\Phi_{tq} = \frac{\partial\Phi_q}{\partial t}$, $\Phi_{tt} = \frac{\partial^2\Phi}{\partial t^2}$ and

$$H = \begin{bmatrix} \sum_i \Phi^1_{q_1 q_i}\dot{q}_i & \cdots & \sum_i \Phi^1_{q_n q_i}\dot{q}_i \\ \vdots & \vdots & \vdots \\ \sum_i \Phi^m_{q_1 q_i}\dot{q}_i & \cdots & \sum_i \Phi^m_{q_n q_i}\dot{q}_i \end{bmatrix}.$$

We now show:

Theorem [10]. *Consider the ranking $\prec$ defined by $q \prec \dot{q} \prec \lambda \prec \ddot{q} \prec \dot{\lambda} \prec \dddot{q} \prec \cdots$ where the dependent variables q, λ are ordered lexicographically $q_1 \prec q_2 \prec \cdots$ and $\lambda_1 \prec \lambda_2 \prec \cdots$. The system* (7), (8), (9), (10) *is in implicit* RIF*-form with A, b, $[r_1, \ldots, r_k]^T$ in the definition above given by*

$$A = \begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix}, \quad b = \begin{bmatrix} F(t, q, \dot{q}) \\ -H\dot{q} - 2\Phi_{tq}\dot{q} - \Phi_{tt} \end{bmatrix}, \quad [r_1, \ldots, r_k]^T = \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} \tag{11}$$

and $N = \{\Phi = 0,\ \Phi_q\dot{q} + \Phi_t = 0\}$, $\det(A) \neq 0$.

Applying this to the slider crank, with the ranking $\theta_1 \prec \theta_2 \prec \dot{\theta_1} \prec \dot{\theta_2} \prec \lambda \prec \ddot{\theta_1} \prec \ddot{\theta_2} \prec \dot{\lambda} \prec \cdots$, we obtain the implicit RIF-form for the system:

$$AX = b. \tag{12}$$

Here

$$\begin{aligned} A &= \begin{bmatrix} l_1^2\left(\frac{1}{4}m_1 + m_2 + m_3\right) + J_1 & -l_1\,l_2 M\,\cos\theta_3 & l_1\,\cos\theta_1 \\ -l_1\,l_2 M\,\cos\theta_3 & {l_2}^2\left(\frac{1}{4}m_2 + m_3\right) + J_2 & -l_2\,\cos\theta_2 \\ l_1\,\cos\theta_1 & -l_2\,\cos\theta_2 & 0 \end{bmatrix}; \\ b &= \begin{bmatrix} -l_1\,g\left(\frac{1}{2}m_1 + m_2 + m_3\right)\cos\theta_1 - l_1\,l_2 M\,\dot{\theta_2}^2\sin\theta_3 \\ l_2 M\,g\cos\theta_2 - l_1\,l_2 M\,\dot{\theta_1}^2\sin\theta_3 l_1\,\sin\theta_1\dot{\theta_1}^2 - l_2\,\sin\theta_2\dot{\theta_2}^2 \end{bmatrix}; \\ X^T &= [r_1, r_2, r_3]^T = [\ddot{\theta_1}, \ddot{\theta_2}, \lambda]^T \end{aligned}$$

where for the sake of a compact presentation, we have used $M = \frac{1}{2}m_2 + m_3$ and $\theta_3 = \theta_1 + \theta_2$. The constraints are

$$\Phi = l_1\,\sin\theta_1 - l_2\,\sin\theta_2 = 0, \tag{13}$$

$$D_t\Phi = l_1\,\cos\theta_1\dot{\theta_1} - l_2\,\cos\theta_2\dot{\theta_2} = 0. \tag{14}$$

We note that neither the matrix A, nor vector b include the dependent variable $\lambda(t)$, which was generated by DYNAFLEX. Because of this, the ode system with constraints (12), (13), (14), can be symbolically solved separately for the variables $\theta_1(t)$, $\theta_2(t)$.

4. Explicit Reduced Involutive Form with Large Expression Management

Above, we obtained the implicit RIF-form for the slider crank as (12), (13), (14). We could in principle symbolically invert the matrix A using Maple, and get the explicit RIF-form, but Maple will give a huge output for A^{-1} and require a lot of memory and computation time. Therefore, we have used symbolic inversion using large expression management (LEM). A full discussion of large expression management will be given elsewhere. Here we give a brief outline.

We have used the MAPLE package `LargeExpressions` which is based on tools first developed for perturbation calculations in fluid mechanics [13]. From the paper [13], we have the following definition for a hierarchy and the main idea for hierarchical representations.

Definition [Hierarchy]. A hierarchy is an ordered list $[S_0, S_1, \ldots]$ of symbols, together with an associated list $[D_0, D_1, \ldots]$ of definitions of the symbols. For each $s \in S_i$ with $i \geq 1$, there is a definition $d \in D_i$ of the form $s = f(\sigma_1, \sigma_2, \ldots, \sigma_k)$ where f is some well-understood function such as an elementary function and each σ_j is a symbol in $[S_0, S_1, \ldots, S_{i-1}]$ and is thus lower in the hierarchy than s.

A computation sequence c is recursively defined as an expression of the form $c = g(s_1, s_2, \ldots, s_k)$ containing symbols s_j from a known hierarchy, together with the computation sequences defined by the associated definitions $d_1, d_2, \ldots, d_k$ of the symbols appearing in c. Obviously a computation sequence defined in terms of symbols in S_0, the set of atoms of the system, is just an expression.

Intuitively, a hierarchy is a framework for constructing computation sequences, and a computation sequence is an expression defined in terms of simpler expressions. We used the package `LargeExpressions` to code our own LU symbolic decomposition routine and also the forward and backward substitutions for solving a linear system. The details will be given elsewhere [14]. After LU decomposition with pivoting and zero-recognition, we get matrices L and U with the pivoting P as follows:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{-{l_1}^2\left(\frac{1}{4}m_1+m_2+m_3\right)+J_1}{l_1\, l_2 M\, \cos\theta_3} & 1 & 0 \\ \frac{-\cos\theta_1}{l_2 M\, \cos\theta_3} & \frac{-4W_3}{W_1} & 1 \end{bmatrix}, \qquad P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

$$U = \begin{bmatrix} -l_1\, l_2 M\, \cos\theta_3 & {l_2}^2\left(\frac{1}{4}m_2 + m_3\right) + J_2 & -l_2\, \cos\theta_2 \\ 0 & -\frac{1}{8}W_1 & \frac{1}{2}W_2 \\ 0 & 0 & -2\,W_4 \end{bmatrix}, \tag{16}$$

where $PA = LU$. Now using forward and backward substitution in the usual way, we get X coded with the `LargeExpressions` package.

After that, we get Explicit RIF-form in the computation sequence form:

$$\ddot{\theta}_1 = -W_7, \tag{17}$$

$$\begin{aligned} \ddot{\theta}_2 = \frac{8}{W_1}[&-l_1\, g\left(\frac{1}{2}m_1 + m_2 + m_3\right)\cos\theta_1 \\ &-l_1\, l_2\, \dot{\theta}_2^2 \sin\left(\theta_1 + \theta_2\right)\left(\frac{1}{2}m_2 + m_3\right) - \frac{1}{4}W_5 + \frac{1}{4}W_2 W_6], \end{aligned} \tag{18}$$

$$\lambda = \frac{1}{2}W_6, \tag{19}$$

and with the constraints

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0; \tag{20}$$

$$D_t\Phi = l_1 \cos\theta_1\dot{\theta}_1 - l_2 \cos\theta_2\dot{\theta}_2 = 0. \tag{21}$$

The complete symbolic expressions for the $W[i]$ are given in the appendix.

The equations (17), (18), (19), (20), (21) are also the explicit RIF form for the 2d slider crank. The symbolic preprocessing helps find all the hidden constraints in the general DAE system (1), (2). This preprocessing procedure makes it possible for us to integrate only interesting variables without computing all the independent variables. For example, to this two dimensional slider crank, as already mentioned, now we can symbolically separate (19) containing $\lambda(t)$ from the other (17), (18), (20), (21) and we can solve directly for $\theta_1(t)$ and $\theta_2(t)$.

5. Numerical Integration Using Computation Sequences

We now show how the two second-order (17), (18) for θ_1 and θ_2 can be integrated. In general it is not possible to unveil all the expressions in the array W. To do so would be to introduce the memory problems that were avoided by using the `Veil` command. Instead we set up computation sequences to pass to `dsolve`. We use MAPLE for the calculations. In particular we only use the MAPLE `LargeExpressions` command `Unveil` to a minimal depth 1, to express the computation sequence of substitution rules SW:

```
> SW := [seq(W[i] = Unveil[W](W[i],1), i = 1 .. LastUsed[W])];
```

Then in order to use `dsolve/numeric`, we make the variable substitutions as follows:

> YW := [subs$(\theta_1(t) = Y[1], \theta_2(t) = Y[3], \dot{\theta}_1(t) = Y[2], \dot{\theta}_2(t) = Y[4], SW)$];

Now we use the `codegen` package in Maple to make a procedure for the numeric integration.

```
> f := codegen[makeproc]( YW, YP[1] = Y[2],  YP[2] = rhs(odesys[1]),
YP[3] = Y[4],YP[4] =  rhs(odesys[2]], parameters = [N, t, Y, YP] );
```

In order to complete the demonstration, we integrate the system for sample numerical values. For the parameters, we choose the following values:

$l1 := 1; l2 := 2; m1 := 1; m2 := 1; m3 := 2; g := 9.8; J1 := 4.5; J2 := 5.5;$

For the initial conditions, we choose the following

$$\theta_1(0) = 0,\ \dot{\theta}_1(0) = 1,\ \theta_2(0) = 0,\ \dot{\theta}_2(0) = \frac{1}{2}$$

which are consistent with the constraint equations

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0; \tag{22}$$

$$D_t\Phi = l_1 \cos\theta_1\dot{\theta}_1 - l_2 \cos\theta_2\dot{\theta}_2 = 0. \tag{23}$$

The Maple numeric dsolve routine can solve the system as follows.

```
> ics := array( [0,1,0,1/2] );
> dvars := [θ1(t), θ̇1(t), θ2(t), θ̇2(t)];
> dsol := dsolve(numeric, number=4, procedure =f, start =0,
    initial = ics, procvars = dvars);
```

Since the system is being integrated as an ordinary ODE system rather than a DAE one, we gain in speed but expect to lose accuracy. If we pick up some points for testing the numeric ODE solutions with respect to the algebraic constraints, we see a slow loss of precision. For example, when $t = 1$, we have the solutions *dsol* as $[t = 1., \theta_1(t) = .2629, \dot{\theta}_1(t) = -.4375, \theta_2(t) = .1303, \dot{\theta}_2(t) = -.2130]$, and errors in the constraints (22, 23) are $0.14e - 7$ and $-0.69e - 7$ respectively. When $t = 20$, the errors are $0.42e - 5$ and $0.59e - 6$ respectively.

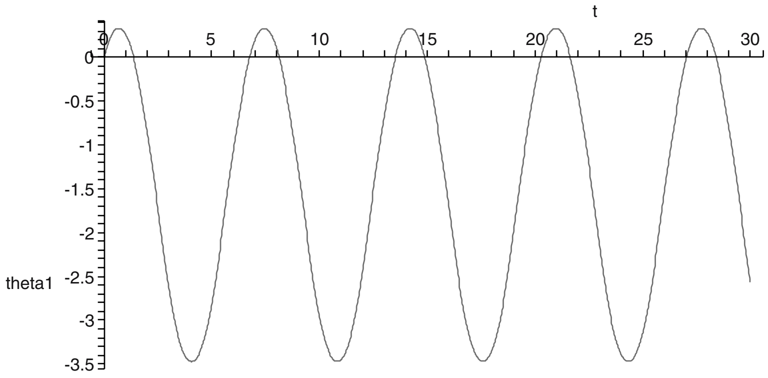


FIGURE 2. The angular θ_1 oscillating movement.

Plots of the numerical simulation of the two dimensional slider crank are shown below as Fig. 2 and Fig. 3. It tells us that the 2d slider crank is moving oscillation with the given initial conditions $\dot{\theta}_1(0) = 1$ and $\dot{\theta}_2(0) = 1/2$. If we increase the initial speeds of θ_1 and θ_2, for example let $\dot{\theta}_1(0) = 2$ and $\dot{\theta}_2(0) = 1$ which are consistent initial conditions, we get Fig. 4 and Fig. 5 which show us the monotonic mode of the angle θ_1.

6. Application of Numerical Algebraic and Analytic Geometry

In this section we discuss the use of some new techniques that are being applied to DAE such as those studied here. A significant problem in large systems, such as the target systems for this work, is the finding of consistent initial conditions. Again there is a need for combined symbolic-numeric methods. The first area that can be harnessed to DAE is that of Numerical Algebraic Geometry [5]. In that approach, components of a polynomial system are characterized by "witness" points, which

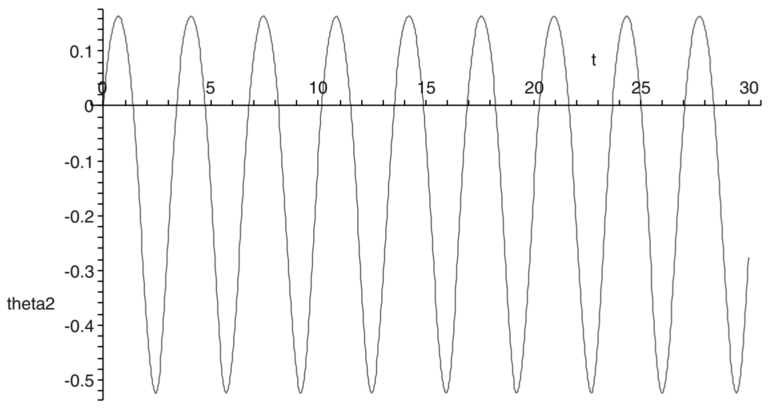


FIGURE 3. The angular θ_2 oscillating movement.

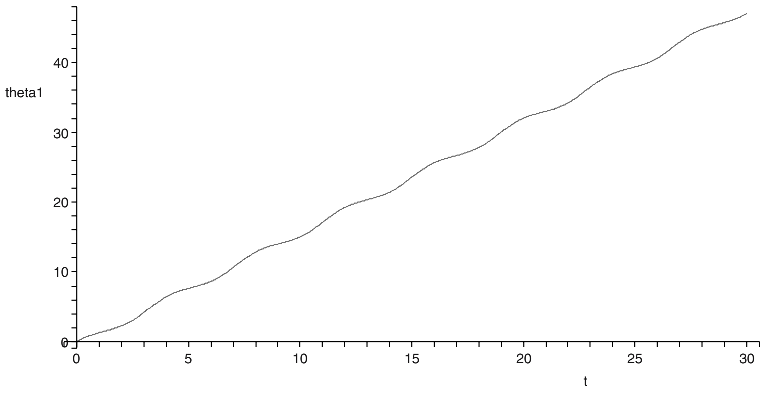


FIGURE 4. Monotonic mode for θ_1.

result from intersecting the component with random linear spaces of complementary dimension. For polynomial DAE, this gives a method of determining points on the constraints, for the consistent initialization of numerical integrators. Also the hybrid symbolic-numeric completion process described in [6] can be applied to polynomial DAE, using numerical algebraic geometry.

Another interesting possibility for analytic DAE is to extend the methods of Numerical Algebraic Geometry [5] as applied to polynomial PDE [6] to the case of analytic DAE. This requires using the substitution of approximate points on the components of the leading nonlinear systems to test ideal membership in the application of RIF. It is natural to generalize the techniques of algebraic geometry

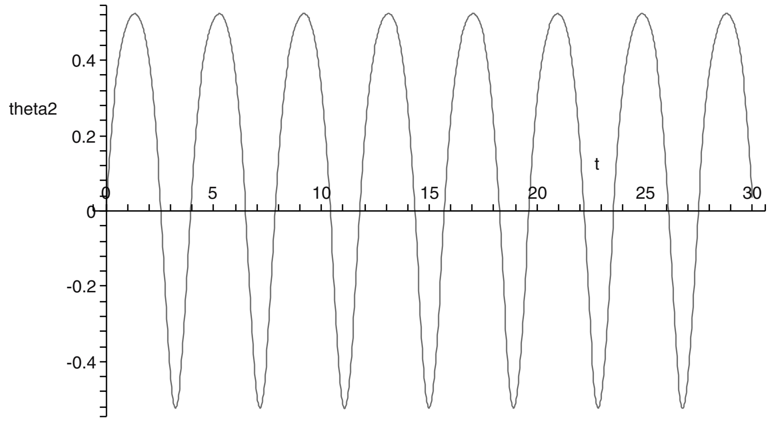


FIGURE 5. The angular motion θ_2 corresponding to monotonic mode of θ_1.

to the analytic case and to characterize irreducible components of analytic functions by points cut out by the intersection of the components with random linear spaces of complementary dimension. The systems are regularized by embedding in appropriate square systems, as in the polynomial case. Newton methods converge to the points on the components, locally. See [11, §5.2.3] for material on the range of non-constant entire functions. Global results like those in the polynomial case, using homotopy continuation, are much harder to obtain. Usually computations must be executed locally on some compact set.

For example, consider the determination of points on the components of an analytic function $f(z, w) = 0$ on some compact set $U \subset \mathbb{C}^2$. By intersecting the component with a random line $\alpha z + \beta w + \gamma = 0$, we obtain a univariate problem whose solutions can be found on $\mathbb{C}$. Powerful tools are already available to automate this process, for example the Maple command `Rootfinding[Analytic]`. This uses the Cauchy Integral Formula to determine the number of zeros in a given sub-domain of $\mathbb{C}$. This subdomain is then divided to isolate the roots. We note that α-theoretic methods can lead to guaranteed convergence, in the isolated zero case, provided certain local criteria are met [9, 8]. A forthcoming work will be devoted to such *Numerical Analytic Geometry* techniques.

7. Conclusion

This paper uses a simple multibody dynamic system to show how we can combine symbolic methods and numeric methods for simulating mechanical systems described by higher-index DAE systems. An advantage of this combination is not only that it helps the numeric method to find consistent initial conditions, but it also extend the potential for symbolic methods, such as RifSimp, to solve more complex symbolic multibody dynamic systems. Using computation sequences to invert a symbolic matrix allows the calculation to be completed in less memory and less time, more details being given in a upcoming paper [14].

We also discussed applying new methods from Numerical Algebraic Geometry and Numerical Analytic Geometry to DAE. Already in her analysis of analytic DAE, Ilie used computation sequences to establish polynomial cost methods. The method upon which she based her complexity analysis is Pryce's structural analysis of DAE [16, 17, 18]. Specifically Pryce's method can be regarded as an efficient way to obtain implicit RifSimp forms, in certain cases. Numerical analytic geometry naturally partners such methods.

Acknowledgements. We thank Silvana Ilie for discussions.

References

[1] P. Shi, J. McPhee. *Symbolic Programming of a Graph-Theoretic Approach to Flexible Multibody Dynamics.* Mechanics of Structures and Machines, **30**(1), 123–154 (2002).

[2] P. Shi, J. McPhee. *Dynamics of Flexible Multibody Systems Using Virtual Work and Linear Graph Theory.* Multibody System Dynamics, **4**(4), 355–381 (2000).

[3] W. Schiehlen. *Multibody Systems Handbook.* Springer-Verlag, Berlin, 1990.

[4] H. Stetter. *Numerical Polynomial Algebra.* SIAM, 2005.

[5] A.J. Sommese, C.W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science.* World Scientific Press, Singapore, 2005.

[6] G. Reid, J. Verschelde, A.D. Wittkopf, W. Wu. *Symbolic-Numeric Completion of Differential Systems by Homotopy Continuation.* Proc. ISSAC 2005, 269–276. ACM Press, 2005.

[7] G. Reid, A. Wittkopf, A. Boulton. *Reduction of Systems of Nonlinear Partial Differential Equations to Simplified Involutive Forms.* Eur. J. Appl. Math., **7**, 604–635 (1996).

[8] M. Giusti, G. Lecerf, B. Salvy, J.-C. Yakoubsohn. *On Location and Approximation of Clusters of Zeros: Case of Embedding Dimension One.* To appear in Foundations of Computational Mathematics, 2006.

[9] M. Giusti, G. Lecerf, B. Salvy, J.-C. Yakoubsohn. *Location and Approximation of Clusters of Zeros of Analytic Functions.* Foundations of Computational Mathematics, **5**, 257–311 (2005).

[10] W. Zhou, D.J. Jeffrey, G.J. Reid, C. Schmitke, J. McPhee. *Implicit Reduced Involutive Forms and Their Application to Engineering Multibody Systems.* Proc. IWMM 2004, LNCS 3519, 31–43. Springer-Verlag, Berlin, 2005.

[11] T. Nishino. *Function Theory in Several Complex Variables.* Translations of Mathematical Monographs 193, AMS, 2001.

[12] R.M. Corless. *Essential Maple 7.* Springer-Verlag, Berlin, 2002.

[13] R.M. Corless, D.J. Jeffrey, M.B. Monagan, Pratibha. *Two Perturbation Calculation in Fluid Mechanics Using Large-Expression Management.* J. Symbolic Computation, **11**, 1–17 (1996).

[14] W. Zhou, D.J. Jeffrey. *LU Symbolic Decomposition with Large Expression Management Strategies.* In preparation.

[15] M.B. Monagan. *Gauss: a Parameterized Domain of Computation System with Support for Signature Functions.* Proc. DISCO '93, LNCS 722, 81–94. Springer-Verlag, Berlin, 1993.

[16] S. Ilie, R.M. Corless, G. Reid. *Numerical Solutions of Index-1 Differential Algebraic Equations Can Be Computed in Polynomial Time.* Numerical Algorithms, **41**, 161–171 (2006).

[17] S. Ilie. *Computational Complexity of Numerical Solutions of Initial Value Problems for Differential Algebriac Equations.* PhD thesis, University of Western Ontario, 2005.

[18] R.M. Corless, S. Ilie. *Polynomial Cost for Solving IVP for High-Index DAE.* Submitted.

[19] J.D. Pryce. *A Simple Structural Analysis Method for DAEs.* BIT **41**(2), 364–394 (2001).

[20] J.D. Pryce. *Solving High-index DAEs by Taylor Series.* Numer. Algorithms, **19**, 195–211 (1998).

Appendix

We append the source listing of the right-hand side of the differential equation, to show the result of Maple's automatic code generation.

$f := \mathrm{proc}(N, t, Y, YP)$

$W[1] := (-4l_1^2l_2^2\cos(Y[1]+Y[3])^2m_2^2 - 16l_1^2l_2^2\cos(Y[1]+Y[3])^2m_2m_3 - 16l_1^2l_2^2 \cos(Y[1]+Y[3])^2m_3^2 + l_1^2m_1l_2^2m_2 + 4l_1^2m_1l_2^2m_3 + 4l_1^2m_1J_2 + 4l_1^2m_2^2l_2^2 + 20l_1^2m_2 l_2^2m_3 + 16l_1^2m_2J_2 + 16l_1^2m_3^2l_2^2 + 16l_1^2m_3J_2 + 4J_1l_2^2m_2 + 16J_1l_2^2m_3 + 16J_1J_2)/(l_1 l_2\cos(Y[1]+Y[3])(m_2+2m_3));$

$W[2] := (-2l_1^2\cos(Y[1])\cos(Y[1]+Y[3])m_2 - 4l_1^2\cos(Y[1])\cos(Y[1]+Y[3])m_3 + \cos(Y[3])l_1^2m_1 + 4\cos(Y[3])l_1^2m_2 + 4\cos(Y[3])l_1^2m_3 + 4\cos(Y[3])J_1)/(l_1 \cos(Y[1]+Y[3])(m_2+2m_3));$

$W[3] := (-2l_2^2\cos(Y[3])\cos(Y[1]+Y[3])m_2 - 4l_2^2\cos(Y[3])\cos(Y[1]+Y[3])m_3 + \cos(Y[1])l_2^2m_2 + 4\cos(Y[1])l_2^2m_3 + 4\cos(Y[1])J_2)/(l_2\cos(Y[1]+Y[3])(m_2 +2m_3));$

$W[4] := (-\cos(Y[1])\cos(Y[3])W[1] + W[3]W[2]\cos(Y[1]+Y[3])m_2 + 2W[3]W[2] \cos(Y[1]+Y[3])m_3)/(\cos(Y[1]+Y[3])(m_2+2m_3)W[1]);$

$W[5] := (-g\cos(Y[3]) + l_1Y[2]^2\sin(Y[1]+Y[3]))(l_1^2m_1 + 4l_1^2m_2 + 4l_1^2m_3 + 4J_1) /(\cos(Y[1]+Y[3])l_1);$

$W[6] := (-l_1 \sin(Y[1])Y[2]^2 \cos(Y[1]+Y[3])W[1]+l_2 \sin(Y[3])Y[4]^2 \cos(Y[1]$
$+Y[3])W[1]-\cos(Y[1])W[1]g\cos(Y[3])+\cos(Y[1])W[1]l_1Y[2]^2 \sin(Y[1]+Y[3])$
$-2W[3] \cos(Y[1]+Y[3])l_1g\cos(Y[1])m_1-4W[3] \cos(Y[1]+Y[3])l_1g\cos(Y[1])m_2$
$-4W[3] \cos(Y[1]+Y[3])l_1g\cos(Y[1])m_3-2W[3] \cos(Y[1]+Y[3])l_1l_2Y[4]^2 \sin(Y[1]$
$+Y[3])m_2-4W[3] \cos(Y[1]+Y[3])l_1l_2Y[4]^2 \sin(Y[1]+Y[3])m_3-W[3] \cos(Y[1]$
$+Y[3])W[5])/(W[4]W[1] \cos(Y[1] + Y[3]));$

$W[7] := (-l_2g\cos(Y[3])W[1]m_2-2l_2g\cos(Y[3])W[1]m_3+l_1l_2Y[2]^2 \sin(Y[1]$
$+Y[3])W[1]m_2+2l_1l_2Y[2]^2 \sin(Y[1]+Y[3])W[1]m_3-2l_2^2m_2l_1g\cos(Y[1])m_1-4l_2^2$
$m_2^2l_1g\cos(Y[1])-20l_2^2m_2l_1g\cos(Y[1])m_3-2l_2^3m_2^2l_1Y[4]^2 \sin(Y[1]+Y[3])-12l_2^3$
$m_2l_1Y[4]^2 \sin(Y[1]+Y[3])m_3-l_2^2m_2W[5]-l_2^2m_2W[2]W[6]-8l_2^2m_3l_1g\cos(Y[1])$
$m_1-16l_2^2m_3^2l_1g\cos(Y[1])-16l_2^3m_3^2l_1Y[4]^2 \sin(Y[1]+Y[3])-4l_2^2m_3W[5]-4l_2^2m_3$
$W[2]W[6] - 8J_2l_1g\cos(Y[1])m_1 - 16J_2l_1g\cos(Y[1])m_2 - 16J_2l_1g\cos(Y[1])m_3$
$-8J_2l_1l_2Y[4]^2 \sin(Y[1]+Y[3])m_2-16J_2l_1l_2Y[4]^2 \sin(Y[1]+Y[3])m_3-4J_2W[5]$
$-4J_2W[2]W[6]+l_2\cos(Y[3])W[6]W[1])/(W[1]l_1l_2\cos(Y[1]+Y[3])(m_2+2m_3));$

$YP[1] := Y[2]; YP[2] := W[7]; YP[3] := Y[4]; YP[4] := 8(-l_1g(1/2m_1+m_2+m_3)$
$\cos(Y[1])-l_1l_2Y[4]^2 \sin(Y[1]+Y[3])(1/2m_2+m_3) - 1/4W[5]$
$-1/4W[2]W[6])/W[1];$

endproc

Wenqin Zhou, David J. Jeffrey and Greg J. Reid
Department of Applied Mathematics
The University of Western Ontario
Middlesex College
1151 Richmond St. N.,
London, Ontario, Canada N6A 5B7
e-mail: `wzhou7@uwo.ca`
`djeffrey@uwo.ca`
`reid@uwo.ca`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 349–359
© 2007 Birkhäuser Verlag Basel/Switzerland

A Symbolic-Numeric Approach to an Electric Field Problem

David J. Jeffrey, Silvana Ilie, James M. Gardiner and Steven W. Campbell

Abstract. A combination of symbolic and numerical methods is used to extend the reach of the purely symbolic methods of physics. One particular physics problem is solved in detail, namely, a computation of the electric potential in the space between a sphere and a containing cylinder. The potential is represented as an infinite sum of multipoles, whose coefficients satisfy an infinite system of linear equations. The system is solved first symbolically by using a series expansion in a critical ratio, namely the ratio of the sphere radius to cylinder radius. Purely symbolic methods, however, cannot complete the solution for two reasons. First, the coefficients in the series expansion can only be found numerically, and, second, the convergence rate of the series is too slow. The combination of symbolic and numerical methods allows the singular nature of an important special case to be identified.

Mathematics Subject Classification (2000). Primary 35C99; Secondary 68W25; Tertiary 65B99.

Keywords. Laplace equation, series solution, asymptotic solution, convergence, numerical analysis, symbolic analysis.

1. Introduction

Many problems in theoretical physics are solved using purely symbolic methods. Such problems, however, are usually restricted to simple situations, for example, a sphere falling slowly through an infinite fluid, or flow past a two-dimensional airfoil. In contrast, symbolic methods have little success with more realistic problems. For example, consider the problem of trying to calculate the flow of a fluid around an object in a tube. The traditional symbolic methods of fluid mechanics were applied by Happel and Brenner [2], who made a lengthy symbolic, but approximate, calculation for a small sphere inside a much larger tube. The problem of a sphere of a reasonable size cannot be solved accurately by their methods. In addition, their calculation was restricted to particles of spherical shape. With these observations in mind, supporters of purely numerical methods argue that symbolic methods are not capable of contributing efficiently to the solution of flow problems like these. This implies that computer systems such as Maple, which facilitate symbolic manipulation, are equally incapable of contributing to flow problems. Further, given progress in the automatic generation of computational grids and computational schemes for numerical methods, there is a danger that symbolic methods will be largely pushed out of many areas of physics and engineering, and remain present only in the initial set up of a problem.

In response to this situation, we present here a symbolic–numeric approach to one class of physics problems, which we describe with the aid of a specific problem, namely the electric field around a non-conducting spherical bubble in a cylindrical wire, a problem first considered in [1]. The starting point is an expansion in eigenfunctions, a classical method of 19th century physics. By itself this fails because the coefficients cannot be found in closed form. Therefore one tries to compute them numerically. This is still not very successful because the series converge very slowly and further analysis is needed. By combining a numerical calculation of the coefficients with a symbolic analysis of the series, we arrive at a useful expression for the resistance of the wire.

The new method offers advantages both when compared with purely numerical methods, and when compared with purely symbolic methods. On the one hand, when compared with purely symbolic methods, the present methods get a solution, which otherwise is beyond reach. On the other hand, when compared with numerical solutions, the present solution retains symbolic information, and moreover very useful information. Specifically, there is a ratio of lengths in the specification of the problem, namely the ratio of the diameter of the spherical bubble to the diameter of the tube, and this ratio is present as a symbolic parameter, which means that the solution obtained is valid for all ratios, whereas a numerical solution requires a complete repetition of the solution procedure. A further advantage of the symbolic–numeric approach is the fact that the problem contains a singular limit. When the bubble nearly fills the tube, i.e., when the ratio of diameters approaches 1, there can be singular effects. The presence of a symbolic parameter in the solution allows the singular behaviour to be studied analytically.

Another aspect of the problem should be noted. The information required from the solution to the problem has an important influence on the solution technique. Here we are interested in the effect of the bubble on the resistance of the wire. Thus we need to calculate one quantity, namely the additional resistance or equivalently the increased effective length. It turns out that this quantity can be extracted neatly from the solution. If we had been interested in something different, for example, the details of the electric field around the bubble, the current method may be of less interest. It is one of a number of techniques being developed to extend the role of symbolic computation in Science and Engineering.

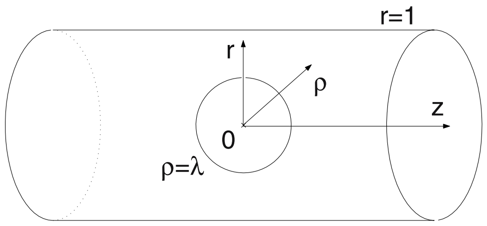


FIGURE 1. The coordinate systems for the sphere inside the cylinder.

Laplace's equation described the electric potential associated with an electric field, and has been studied in the space outside a sphere and inside a cylinder by Linton [5]. Linton used the method of multipoles to derive a solution of the problem. His technique consists of first constructing a set of functions that satisfy the equation and all the boundary conditions except the one on the sphere, and then representing the solution as a superposition of all the functions in the set. Satisfying the condition on the sphere leads to an infinite set of linear equations in the coefficients of the solution. The new feature of the present paper consists in expressing the unknown coefficients as series expansions in the geometrical parameter referred to above: the ratio of the sphere diameter to the cylinder diameter. The solution is exact for all diameter ratios, but its rate of convergence slows down in the limit of the particle blocking the tube.

When the particle nearly blocks the tube, a different approach can be used. An asymptotic analysis allows us to solve the problem approximately. By comparing the general solution and the asymptotic solution, we predict the behaviour of the coefficients in the general solution, and thereby improve its rate of convergence.

2. Solution for All Diameter Ratios

We consider the electric field present in a cylindrical tube of radius d containing a sphere of radius a situated on the axis. The field is produced by a potential gradient, which causes a current to flow through the tube; equivalently, the electric field tends to a constant at infinity. We use the cylindrical coordinate system (r, z, ϕ) and the spherical coordinate system (ρ, θ, ϕ) which both have their origins at the centre of the sphere and are rescaled so that the cylinder boundary corresponds to $r = 1$. The spherical boundary is then given by $\rho = \lambda$ where λ is the ratio of the sphere radius to the cylinder radius. The axial symmetry is used to suppress reference to the azimuthal angle ϕ. The two systems of coordinates are connected by the relations $z = \rho\cos\theta,\ r = \rho\sin\theta$.

We introduce an electric potential of the form $z + \Phi$. Because of the linearity of Laplace's equation, we can scale the potential so that the electric field tends to unity at infinity. The disturbance potential Φ must satisfy Laplace's equation in cylindrical coordinates

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \Phi}{\partial r}\right)+\frac{\partial^2\Phi}{\partial z^2}=0\ ,$$

together with the boundary conditions

$$\frac{\partial\Phi}{\partial r} = 0\ , \qquad \text{on}\ \ r=1\ , \tag{2.1}$$

$$\frac{\partial\Phi}{\partial \rho} = -\cos\theta\ , \qquad \text{on}\ \ \rho=\lambda\ , \tag{2.2}$$

$$\frac{\partial\Phi}{\partial z} \to 0\ , \quad \text{as} \quad |z|\to\infty\ . \tag{2.3}$$

The last equation is not equivalent to $\Phi\to 0$ as might be expected, but rather to

$$\Phi\to\alpha\,\mathrm{sgn}\,z\ , \quad \text{as} \quad |z|\to\infty\ , \tag{2.4}$$

where α depends on λ and is in general non-zero [4]. The consequences of this for the convergence of the series used below are discussed in [4].

To solve the problem above we use dual expansions. First we express the solution in cylindrical coordinates as a linear combination of functions satisfying the equation and the boundary conditions (2.1) and (2.4). Then we transform the solution into spherical coordinates and apply the condition (2.2).

Starting in cylindrical coordinates, we expand the perturbed potential as a series by using the usual separation of variables. Thus assuming $\Phi = R(r)Z(z)$ (the azimuthal angle ϕ does not enter because of axisymmetry), we find that $R(r)$ is a linear combination of Bessel functions $K_0(tr)$ and $I_0(tr)$, where t^2 is the separation constant. Similarly $Z(z)$ is a combination of $\sin tz$ and $\cos tz$. To satisfy boundary condition (2.1), we must combine the Bessel functions according to $K_0(tr)+[K_1(t)/I_1(t)]I_0(tr)$. Symmetry in z requires that the $\cos tz$ term is dropped. Then integrating over all values of t, we obtain

$$\Phi=\sum_{n=1}^{\infty}\frac{A_n}{2n}\Phi_n \tag{2.5}$$

where

$$\Phi_n=\frac{2\lambda^{2n+1}(-1)^{n+1}}{\pi(2n-1)!}\int_0^\infty t^{2n-1}\left(K_0(tr)+\frac{K_1(t)}{I_1(t)}I_0(tr)\right)\sin(tz)dt\ . \tag{2.6}$$

The following identities allow the transformation between the potential in cylindrical coordinates (r,z) and in spherical coordinates (ρ,θ):

$$\frac{1}{\rho^{2n}}P_{2n-1}(\cos\theta) = \frac{2(-1)^{n+1}}{\pi(2n-1)!}\int_0^\infty t^{2n-1}K_0(tr)\sin(tz)dt\ ,$$

$$I_0(tr)\sin(tz) = \sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{(2n-1)!}(t\rho)^{2n-1}P_{2n-1}(\cos\theta)\ .$$

By applying these identities we deduce

$$\begin{aligned}\Phi &= \sum_{n=1}^{\infty} \frac{A_n}{2n}\frac{\lambda^{2n+1}}{\rho^{2n}} P_{2n-1}(\cos\theta) \\ &+ \sum_{m,n=1}^{\infty} \frac{2A_n(-1)^{n+m}}{\pi(2n)!(2m-1)!}\lambda^{2n+1}\rho^{2m-1}P_{2m-1}(\cos\theta)\int_0^\infty t^{2(n+m-1)}\frac{K_1(t)}{I_1(t)}dt\ .\end{aligned}$$

From the boundary condition (2.2) and the orthogonality of the Legendre functions we obtain an infinite system of algebraic equations for A_n

$$A_n + \sum_{m=1}^{\infty} \lambda^{2n+2m-1} A_m B_{mn} = \delta_{1n} \qquad (n = 1, 2, \ldots) \tag{2.7}$$

with the coefficients B_{mn} given by

$$\begin{aligned}B_{mn} &= \frac{2(-1)^{n+m+1}}{\pi(2m)!(2n-2)!}\int_0^\infty t^{2(n+m-1)}\frac{K_1(t)}{I_1(t)}dt \\ &= \frac{2(-1)^{n+m+1}}{\pi(2m)!(2n-2)!(2n+2m-1)}\int_0^\infty \frac{t^{2(n+m-1)}}{I_1^2(t)}dt\ .\end{aligned}$$

This set of equations was obtained in [1] before computer algebra was readily available. However, the derivation was checked using Maple, but a completely automatic program was not written. To solve the equations, one could truncate them and tackle them numerically at this point, but to do so would be to return to a purely numerical solution, and then it would be dubious whether all of the manipulations above were worthwhile, or whether a finite-element scheme would not be just as good. Therefore, we solve the system of equations by expressing each coefficient as a series in λ; this choice is suggested by the appearance of the term $\lambda^{2n+2m-1}$ in (2.7).

$$A_n(\lambda) = \sum_{s=0}^{\infty} K_{ns}\lambda^s\ . \tag{2.8}$$

Substituting this into (2.7) and collecting powers of λ, we obtain a recurrence relation for the K_{np} coefficients:

$$K_{np} + \sum_{s=1}^{(p+1-2n)/2} K_{s(p+1-2s-2n)}B_{sn} = 0\ , \quad \text{for } n \geq 1, \quad p \geq 2n+1\ , \tag{2.9}$$

and

$$\begin{aligned}K_{n0} &= \delta_{n1}\ , \quad \text{for } n \geq 1 \\ K_{np} &= 0\ , \quad \text{for } n \geq 1,\ 2n \geq p \geq 1.\end{aligned}$$

The numerical solution of (2.9) for the numbers K_{np} is straightforward. Notice that although the problem contains the parameter λ, these are pure numbers independent of λ. In order to speed up the computation and to improve its reliability, we can integrate the dominant contribution to the integral symbolically. Thus, we write

$$\int_0^\infty \frac{t^{2(n+m-1)}}{I_1^2(t)}\,dt = \frac{\pi(2n-1)!}{2^{2n-1}} + \int_0^\infty t^{2(n+m-1)}\left[I_1^{-2}(t) - 2\pi t e^{-2t}\right]\,dt\ .$$

It can be remarked that supporters of computer algebra frequently say that their systems are not purely symbolic, but numeric as well. This aspect of Maple made the calculation of the coefficients very easy because the integrals could be evaluated easily using Maple's numerical integration.

It was stated in the introduction that the solution, which now has been presented, is attractive because we want only the effective resistance to an electric current flowing past the bubble. According to (2.4), the difference in potential increases by the amount 2α, which is *a priori* unknown. By considering the asymptotic behaviour of the integral in (2.6), we find that for $n = 1$ it is asymptotic to $\operatorname{sgn} z$ and for $n > 1$ it tends to 0 (details in [4]). Therefore α depends only on A_1. The full analysis gives

$$\Delta\Phi = 2\alpha = 2\lambda^3 A_1\ . \tag{2.10}$$

Thus the resistance depends only on the single coefficient A_1, and therefore will be expressed as a single series with computable coefficients. The above solution has the advantage that a single computation of the system (2.9) solves the problem for all λ.

An apparent disadvantage is the fact that the series diverges at $\lambda = 1$, which corresponds to the sphere filling the tube. For values of λ near the radius of convergence, the rate of convergence of (2.8) becomes very slow and many terms is needed in the series. However, this effect has a physical basis: the problem is singular when the sphere fills the tube, and even purely numerical methods struggle in this case. This apparent disadvantage, however, can be turned into an advantage in the context of symbolic-numeric computation. We can now analyze the singular limit and match the limit to the general solution. The result is a new series that converges everywhere numerically and which displays the singular behaviour symbolically. This is only possible in a symbolic-numeric context.

Asymptotic Behaviour

We want to study the asymptotic nature of the solution when the sphere is almost the same diameter as the cylinder. The gap between the sphere and the cylinder is measured by the non-dimensional parameter $\varepsilon = 1 - \lambda$ which is assumed much smaller than 1.

We proceed by considering a stretching transformation of the cylindrical coordinates in the gap, based on the physical fact that the effects across the small gap dominate the effects along it. A similar idea in rather different geometry [3] suggests that the proper transformation is

$$R = (1-r)/\varepsilon, \quad Z = z/\sqrt{\varepsilon}.$$

The surface of the sphere in the new coordinates has the expansion

$$R = 1 + \frac{1}{2}Z^2 + \varepsilon\left(\frac{Z^2}{2} + \frac{Z^4}{8}\right) + O(\varepsilon^2).$$

We will apply the boundary condition on this approximation rather than on the exact surface, since the exact expression contains square roots.

The scaled Laplace's equation inside the gap is given by

$$\frac{\partial^2\Phi}{\partial R^2} + \varepsilon\left(\frac{\partial^2\Phi}{\partial Z^2} - \frac{1}{1-\varepsilon R}\cdot\frac{\partial\Phi}{\partial R}\right) = 0 \tag{2.11}$$

and the boundary condition on the cylinder by

$$\frac{\partial\Phi}{\partial R} = 0 \qquad \text{on} \;\; R = 0.$$

In order to deduce the boundary condition on the obstacle we notice that

$$\sin\theta = \frac{1-\varepsilon R}{1-\varepsilon}, \qquad \cos\theta = \frac{\sqrt{\varepsilon}Z}{1-\varepsilon} \qquad \text{on} \;\; \rho = 1-\varepsilon$$

and derive the scaled boundary condition on the sphere

$$\frac{\partial\Phi}{\partial R} - \varepsilon\left(R\frac{\partial\Phi}{\partial R} + Z\frac{\partial\Phi}{\partial Z}\right) = \varepsilon^{3/2}Z. \tag{2.12}$$

We apply Gauss' theorem between $z \to -\infty$ and $z = 0$ and find that $\Phi = O(\varepsilon^{-1/2})$ in the gap. Consequently, it is natural to look for an expansion of the potential of the form

$$\Phi(R,Z) = \varepsilon^{-\frac{1}{2}}\Phi_0(R,Z) + \varepsilon^{\frac{1}{2}}\Phi_1(R,Z) + \cdots.$$

The unknown functions Φ_n are derived by replacing the above expansion in the problem (2.11)-(2.12) and by matching the expansions inside and outside the gap. The first approximation leads to the following problem for Φ_0

$$\begin{aligned}
\frac{\partial^2\Phi_0}{\partial R^2} &= 0, \\
\frac{\partial\Phi_0}{\partial R} &= 0 \qquad \text{on} \;\; R = 0, \\
\frac{\partial\Phi_0}{\partial R} &= 0 \qquad \text{on} \;\; R = 1 + \frac{Z^2}{2},
\end{aligned}$$

which has the solution $\Phi_0 = K(Z)$. The next step in the approximation gives

$$\begin{aligned}
\frac{\partial^2\Phi_1}{\partial R^2} &= -K''(Z), \\
\frac{\partial\Phi_1}{\partial R} &= 0 \qquad\qquad \text{on} \;\; R = 0, \\
\frac{\partial\Phi_1}{\partial R} &= ZK'(Z) \qquad \text{on} \;\; R = 1 + \frac{Z^2}{2}.
\end{aligned} \tag{2.13}$$

From (2.13) we find that K must satisfy

$$(1 + \frac{1}{2}Z^2)K'' + ZK' = 0$$

and using the antisymmetry with respect to Z, we derive $\Phi_0 = C \cdot \arctan(Z/\sqrt{2})$. The constant C should be obtained from a matching with the solution outside the gap, but that would require deriving a solution in the 'outer' region beyond the gap. We can avoid this by using Gauss' theorem to determine the constant C:

$$\Phi = \frac{1}{\sqrt{2\varepsilon}} \cdot \arctan \frac{Z}{\sqrt{2}} + O(\varepsilon^{1/2}). \tag{2.14}$$

Matching General Solution with Asymptotic Solution

The asymptotic solution (2.14) behaves asymptotically like $\operatorname{sgn} z$ with respect to z. Therefore, from (2.10), we obtain

$$A_1 = \frac{\pi}{2\sqrt{2}}(1-\lambda)^{-1/2} + O((1-\lambda)^{1/2}).$$

Thus, if we expand the asymptotic solution with respect to λ, we obtain the series

$$A_1 = \frac{\pi}{2\sqrt{2}} \sum_{p=1}^{\infty} (-1)^p \binom{-1/2}{p} \lambda^p + O((1-\lambda)^{1/2}).$$

We also made the assumption for the general solution that the coefficient expands as (2.8). By matching the two forms of the solution we get the prediction

$$\frac{K_{1p}}{\frac{\pi}{2\sqrt{2}}(-1)^p\binom{-1/2}{p}} \to 1 \qquad \text{as} \quad p \to \infty \ .$$

This prediction is tested in Fig. 2, where it can be seen that the agreement is very good after about 50 terms. A selection of the same data expressed in tabular form is given in Table 1.

The application that suggested this calculation is the change in the electrical resistance of a wire owing to impurities in the metal, and for this only the coefficient A_1 is of interest. The extra resistance is often expressed as an effective increase in the length of the cylinder, and this extra length is given by

$$\Delta L = 2\lambda^3 A_1.$$

Since we know the singular behaviour symbolically, we can extract it from the numerical coefficients and obtain a more reliable calculation. We write

$$\begin{aligned} A_1 &= \sum_{n=0}^{\infty} K_{1n}\lambda^n && (2.15)\\ &= \frac{\pi}{\sqrt{8}(1-\lambda)^{1/2}} + \sum_{n=0}^{\infty}\left[K_{1n} - \frac{\pi}{\sqrt{8}}(-1)^n\binom{-1/2}{n}\right]\lambda^n. && (2.16)\end{aligned}$$

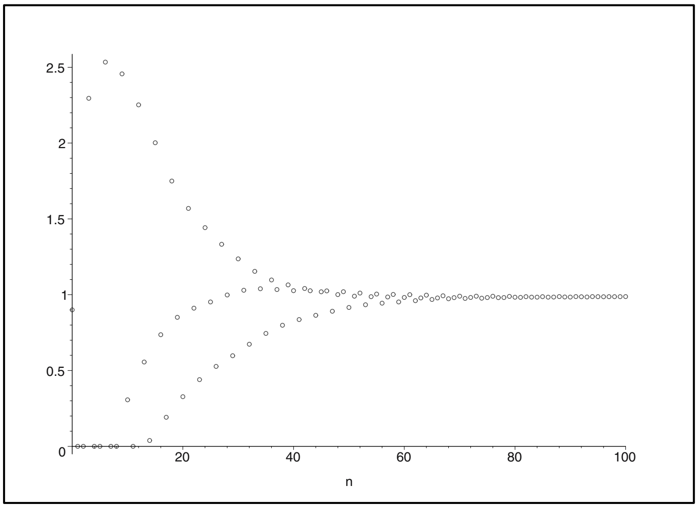


FIGURE 2. The ratio of the coefficients K_{1n} as computed from (2.9) to the predicted value $\frac{\pi}{2\sqrt{2}}(-1)^n\binom{-1/2}{n}$, for different values of n.

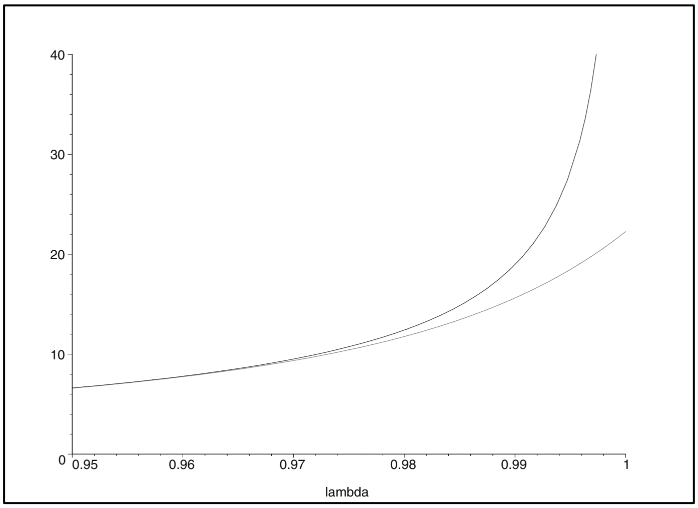


FIGURE 3. The computed resistance increase as a function of λ for values close to 1. The upper curve shows the correct singular behaviour which is expressed symbolically; the lower curve shows the purely numerical result.

Table 1. A comparison of the computed and predicted coefficients in the series for the increased resistance of a conducting cylinder due to the presence of a spherical bubble.

n	K_{1n}	$(-1)^n\pi\binom{-1/2}{n}/\sqrt{8}$	Difference
95	0.063370	0.064209	-0.000839
96	0.062993	0.063876	-0.000883
97	0.062742	0.063547	-0.000805
98	0.062402	0.063222	-0.000820
99	0.062086	0.062903	-0.000817
100	0.061827	0.062587	-0.000760

In Table 1, the last column shows how much smaller the new difference coefficients are, and Fig. 3 shows the effect of extracting the singularity explicitly on the results.

3. Conclusions

This paper has shown that by combining symbolic and numeric techniques, we can obtain new forms for the solution of problems arising in physics. The method used here can be extended to other equations of theoretical physics, such as Stokes's equations and Helmholtz's equation. The principles, although probably not the detailed method, can be extended to other geometries. In general, purely numerical methods will remain more flexible than the present one, but what has been demonstrated is that when symbolic information can be returned to a solution, there is a gain in numerical accuracy and in our understanding of the solution. One pleasing feature of the current method is that the two solutions used were derived independently, and hence the agreement between the two ways of computing the coefficients is a good check on the correctness of the intermediate working.

References

[1] J. M. Gardiner, *A contained potential problem, with special reference to the resistivity of cavitated α-brass*, M.Sc. thesis, University of Melbourne, 1971.

[2] J. Happel, H. Brenner, *Low Reynolds Number Hydrodynamics*, Prantice-Hall, Englewood Cliffs, 1965.

[3] D. J. Jeffrey, *The temperature field or electric potential around two almost touching spheres*, J. Inst. Math. Appl., vol. 22, pp. 337–351, 1978.

[4] S. Ilie, D. J. Jeffrey, *A note on Laplace's equation inside a cylinder*, Appl. Math. Lett., vol. 18 (1), pp. 55–59, 2005.

[5] C. M. Linton, *Multipole methods for boundary-value problems involving a sphere in a tube*, IMA J. Appl. Math., vol. 55, pp. 187–204, 1995.

[6] W. R. Smythe, *Flow around a spheroid in a circular tube*, Phys. Fluids, vol. 7, pp. 633–7, 1964.

[7] G. N. Watson, *A Treatise on the Theory of Bessel Functions*, Cambridge University Press, Cambridge, 1944.

David J. Jeffrey
Department of Applied Mathematics
University of Western Ontario
1151 Richmond St. N.,
London, Ontario, Canada N6A 5B7
e-mail: `djeffrey@uwo.ca`

Silvana Ilie, James M. Gardiner and Steven W. Campbell
e-mail: `silvana@maths.lth.se`
`jamie_gardiner@iosphere.net.au`
`swcampbe@uwo.ca`

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 361–374
© 2007 Birkhäuser Verlag Basel/Switzerland

Financial Applications of Symbolically Generated Compact Finite Difference Formulae

Jichao Zhao, Robert M. Corless and Matt Davison

Abstract. We introduce the standard fourth order compact finite difference formulae. We show how these formulae apply in the special case of the heat equation. It is well known that the American option pricing problem may be formulated in terms of the Black Scholes partial differential equation (PDE) together with a free boundary condition. Standard methods allow this problem to be transformed into a moving boundary heat equation problem. We use the compact finite difference method to reduce this problem to a system of ordinary differential equations with specified initial conditions. We develop three ways of combining the resulting systems with methods designed to cope with free boundary values. We show that the compact finite difference scheme for the heat equation and for the American options pricing problem are unconditionally stable. After numerical comparison of these methods with a standard Crank Nicholson projected Successive Over Relaxation method, we conclude that the compact finite difference technique respresents an exciting new method for pricing American options.

1. Introduction

In this paper, we first introduce the standard compact finite difference formulae and show how to generate them symbolically. Then we adjust compact finite difference formulae for heat equations. The American option pricing problem, i.e. Black-Scholes equation with free boundary conditions, is converted into ordinary differential equation after we employ compact finite difference method on it. It can be modified to use the built-in ordinary differential equation solvers in many software packages, like Matlab and Maple. We use three different ways (refer to [13, 3]) to deal with free boundary values. Last through comparing with the Crank Nicholson projected Successive Over Relaxation (SOR) method, we know that compact finite difference method converges faster than it under some conditions for American option pricing problems.

2. Symbolic Generation of Compact Finite Difference Formulae

Compact finite difference method is a special finite difference method which uses the values of the function only at three consecutive points to approximate its derivatives at the same three points with high accuracy. Let us focus on one dimension case throughout this paper, first discretize the interval $[a, b]$, let $h = \triangle x = \frac{b-a}{N+1}$, and $x_i = a + ih, i = 0, 1, \ldots, N, N + 1$.

2.1. Standard Compact Finite Difference Formulae

The standard compact finite difference formula of one dimension for second derivatives is

$$\frac{d^2 v_{i-1}}{dx^2} + 10\frac{d^2 v_i}{dx^2} + \frac{d^2 v_{i+1}}{dx^2} = \frac{12(v_{i-1} - 2v_i + v_{i+1})}{h^2}, \tag{2.1}$$

where $v_i = v(x_i)$. The coefficients of the compact finite difference formula for second derivatives can be determined in the following way:

(i) Write down the desired compact finite difference formula with unknown coefficients:

$$a_{-1}\frac{d^2 v_{i-1}}{dx^2} + a_0\frac{d^2 v_i}{dx^2} + a_1\frac{d^2 v_{i+1}}{dx^2} = \frac{(m_{-1}v_{i-1} + m_0 v_i + m_1 v_{i+1})}{h^2}, \tag{2.2}$$

where a_{-1}, a_0, a_1, m_{-1}, m_0, and m_1 are the parameters to be decided later.

(ii) Expand both sides of the Equ. (2.2) in Taylor series at the point x_i with respect to the discretization parameter h, then collect them by the order of h.

(iii) we obtain six equations by setting the coefficients of h^j, $j = -2, -1, \ldots, 2, 3$ equal zero. Solve the six equations for the six unknown parameters. Then we obtain the formula (2.1).

From the above algorithm, we can see easily the accuracy is $O(h^4)$ for formula (2.1). In the appendix, we give Maple codes to generate fourth order compact finite difference schemes for second derivatives, and it can be easily modified for other cases. The standard compact finite difference formula of one dimension for first derivatives is

$$\frac{dv_{i-1}}{dx} + 4\frac{dv_i}{dx} + \frac{dv_{i+1}}{dx} = \frac{3(-v_{i-1} + v_{i+1})}{h}. \tag{2.3}$$

In a similar way, we can obtain the coefficients of compact finite difference formula for first derivatives with fourth order accuracy. Corless, Rokicki and Zhao in [6] made a maple routine for generation of finite difference formulae with any dimensions, which can also generate any form of compact finite difference formulae.

2.2. Adjust Formulae for Unknown Values of Second Derivatives

In this part, suppose that we have known the values of v and want to solve for the unknown v''. We need to adjust the standard compact finite difference formulae for the end points so that the tridiagonal system obtained by writing them into matrix form can be factorized exactly.

Equation (2.1) is used at interior grid points when $i = 2, \ldots, N-1$. When $i = 1$, we have

$$\begin{aligned} cv_1'' + v_2'' &= \frac{1}{12h^2}((10c-1)v_0 - (15c-1)v_1 - 2(2c+15)v_2 \\ &\quad +2(7c+8)v_3 - (6c+1)v_4 + cv_5), \end{aligned} \tag{2.4}$$

where the c is a parameter to be decided later. And when $i = N$

$$\begin{aligned} v_{N-1}'' + 10v_N'' &= \frac{1}{12h^2}(10v_{N-4} - 61v_{N-3} + 156v_{N-2} \\ &\quad -70v_{N-1} - 134v_N + 99v_{N+1}), \end{aligned} \tag{2.5}$$

both of the formulae for end points are not compact, but obtain $O(h^4)$, and can also be obtained by Taylor expansion. Also note the values of v_0 and v_{N+1} in above two equations are supposed to be the known boundary conditions. Write the compact finite difference formulae into the following matrix form:

$$AV'' = MV + H, \tag{2.6}$$

where

$$A = \begin{pmatrix} c & 1 & 0 & \cdots & 0 \\ 1 & 10 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & 10 & 1 \\ 0 & \cdots & 0 & 1 & 10 \end{pmatrix}_{N\times N}, \text{ and } V'' = \begin{pmatrix} v_1'' \\ v_2'' \\ \vdots \\ v_{N-1}'' \\ v_N'' \end{pmatrix},$$

$$M = \frac{12}{h^2}\begin{pmatrix} -\frac{(15c-16)}{144} & -\frac{2(2c+15)}{144} & \frac{2(7c+8)}{144} & -\frac{(6c+1)}{144} & \frac{c}{144} & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & & \vdots \\ 0 & 1 & -2 & 1 & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & \frac{10}{144} & \frac{-61}{144} & \frac{156}{144} & \frac{-70}{144} & \frac{-134}{144} \end{pmatrix},$$

$$H = \frac{1}{12h^2}\begin{pmatrix} (10c-1)v_0 \\ 0 \\ \vdots \\ 0 \\ 99v_{N+1} \end{pmatrix}_{N\times 1} \text{ and } V = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \\ v_N \end{pmatrix}.$$

This is the main idea for compact finite difference method, by using linear combination of the known values of a function at several points to approximate a linear combination of the unknown values of derivatives of the function at several points with high accuracy.

Note that the above formula (2.4) we derived is always true for any value of parameter c, by carefully choosing the c (we choose $c = 5 + 2\sqrt{6}$, refer to [6]) so that matrix A factors exactly into $A = LU = LDL^T$, where

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ k & 1 & 0 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & k & 1 & 0 \\ 0 & \cdots & 0 & k & 1 \end{pmatrix},$$

with $k = 1/c$, and $D = \operatorname{diag}(c, c, \ldots, c)$. Then the solution (use v to express the values of v'') to $AV'' = b$ (where $b = MV + H$) can be solved very efficiently and accurately, and the cost is $O(n)$ flops: by first solving for the vector Y

$$LY = b, \tag{2.7}$$

i.e.

$$y_1 = b_1,\, y_i = b_i - ky_{i-1},\;\; i = 2, \ldots, N, \tag{2.8}$$

where

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix},$$

and then solving for V

$$UV = Y, \tag{2.9}$$

i.e.

$$v_N = ky_N,\, v_i = k(y_i - v_{i+1}),\;\; i = N-1, \ldots, 1, \tag{2.10}$$

the error is damped at each step for the fact that $k < 1$.

So we can solve for unknown values of v'' in term of known values of v with high order of accuracy $O(h^4)$ and $O(N)$ operations by compact finite difference method.

2.3. Compact Finite Difference Method for Heat Equations

We will show how to adjust compact finite difference formulae for the following heat equations:

$$u_\tau(x, \tau) = u_{xx}(x, \tau) + g(x, \tau), \tag{2.11}$$

where $x \in [a, b]$, $t \in (0, T]$ and $g(x, \tau)$ is a given function, heat equation's initial and boundary conditions are

$$u(x, 0) = u_0(x),\ x \in (a, b), \tag{2.12}$$

$$u(a, t) = u_a(t), \tag{2.13}$$

$$u(b, t) = u_b(t) \tag{2.14}$$

and the functions $u_0(x)$, $u_a(t)$ and $u_b(t)$ are already known functions.

We use compact finite difference schemes for the derivative parts $u_{xx}(x, \tau)$ with regard to variable x of the heat Equ. (2.11) as we did in Sect. 2.2. We use v to stand for the approximation value of u throughout this paper, and obtain the following high accurate formulae:

$$\begin{aligned} cv_1''(\tau) + v_2''(\tau) &= \frac{1}{12h^2}((10c-1)v_0(\tau) - (15c-1)v_1(\tau) - 2(2c+15)v_2(\tau) \\ &\quad +2(7c+8)v_3(\tau) - (6c+1)v_4(\tau) + cu_5(\tau)), \end{aligned} \tag{2.15}$$

where we can choose the value of c as we did in Sect. 2.2,

$$\begin{aligned} v_{i-1}''(\tau) + 10v_i''(\tau) + v_{i+1}''(\tau) &= \frac{12}{h^2}(v_{i-1}(\tau) - 2v_i(\tau) + v_{i+1}(\tau)), \\ &\quad i = 2, \ldots, N-1, \end{aligned} \tag{2.16}$$

$$\begin{aligned} v_{N-1}''(\tau) + 10v_N''(\tau) &= \frac{1}{12h^2}(10v_{N-4}(\tau) - 61v_{N-3}(\tau) + 156v_{N-2}(\tau) \\ &\quad -70v_{N-1}(\tau) - 134v_N(\tau) + 99v_{N+1}(\tau)). \end{aligned} \tag{2.17}$$

We can write them in the following compact matrix form:

$$AV''(\tau) = MV(\tau) + H(\tau), \tag{2.18}$$

where

$$V''(\tau) = \begin{pmatrix} v_1''(\tau) \\ v_2''(\tau) \\ \vdots \\ v_{N-1}''(\tau) \\ v_N''(\tau) \end{pmatrix},\ H(\tau) = \frac{1}{12h^2}\begin{pmatrix} (10c-1)v_0(\tau) \\ 0 \\ \vdots \\ 0 \\ 99v_{N+1}(\tau) \end{pmatrix},\ V = \begin{pmatrix} v_1(\tau) \\ v_2(\tau) \\ \vdots \\ v_{N-1}(\tau) \\ v_N(\tau) \end{pmatrix},$$

and the matrix A and M are defined in Sect. 2.2. Then we factor matrix A into LDL^T and solve for the unknown $V''(\tau)$ at the time step τ.

At the end of this part, we explain the procedure to solve the heat equation by compact finite difference method:

(i) Set initial conditions $V(0)$ by Equ. (2.12).
(ii) Suppose we know the the values of $V(\tau^m)$ at time step m, and want to compute for $V(\tau^{m+1})$ at the time step $m + 1$. We solve the system of linear equations $AV''(\tau^m) = b(\tau^m)$ and $b(\tau^m) = MV(\tau^m) + H(\tau^m)$ for $V''(\tau^m)$ by compact finite difference method, say $V''(\tau^m) = \Phi(V^m)$ at each time step. And the vector $H(\tau^m)$ is known since the boundary conditions Equ. (2.13) and

Equ. (2.14) are given. Then the matrix form of Equ. (2.11) can be rewritten as

$$V_\tau = \Phi(V^m) + G(\tau^m), \tag{2.19}$$

where the $G(\tau^m)$ is a vector and its entries are $g(x_i, \tau^m)$, $i = 1, \ldots, N$. The above Equ. (2.19) is just an ordinary differential equation with the initial condition V^m; then we solve the ode problem Equ. (2.19) to obtain V^{m+1} with the accuracy of $O(h^4)$ by Runge-Kutta method. In our implementation, we use Matlab's built-in ode solver to solve Equ. (2.19) with high accuracy and efficiency.

(iii) Repeat step *(ii)* until $\tau^{m+1} = T$.

From the above algorithm, we can solve for the values $u(x,\tau)$ of heat Equ. (2.19) with high order accuracy of $O(\tau^4 + h^4)$.

Remark 2.1. Since the entries of compact finite difference matrix A are constant, we can predecompose matrix A to save time. In other words, we do not need to factor matrix A at all in our codes to solve for heat equations.

3. Compact Finite Difference Method for American Option Pricing

It is well known that American option pricing problem can be transformed into a heat equation with a free moving boundary condition, which makes it much harder to solve. In this section, we develop three ways to combine compact finite difference method for American option pricing with methods especially dealing with free moving boundary conditions.

By using the transformations in [13], we have the following transformed equations for American put option pricing:

$$u_\tau = u_{xx} + g(x,\tau), \tag{3.1}$$

where $x \in (X^*(\tau), +\infty)$, $\tau \in (0, \frac{1}{2}\sigma^2 T]$, σ is a given constant, $X^*(\tau)$ stands for free boundary value at time step τ, and

$$g(x,\tau) = e^{k_1\tau}((k_1 - k_2)e^{x-(k_2-1)\tau)} - k_1),$$

its initial and boundary conditions are

$$u(x,0) = \max(e^x - 1, 0), \tag{3.2}$$

$$X^*(0) = \min(0, \ln(r/D)), \tag{3.3}$$

$$u(X^*(\tau),\tau) = 0, \tag{3.4}$$

$$\lim_{x\to+\infty} u(x,\tau) = e^{k_1\tau}(e^{x-(k_2-1)\tau} - 1), \tag{3.5}$$

where k_1, k_2, r, and D are given parameters.

To solve the American option pricing problem, we need to decide the free boundary value $X^*(\tau)$ at each time step τ. Dependent on how we compute the location of free boundary values, we develop three compact finite difference methods. Compact finite difference method one is to use an implicit condition that the solutions of transformed partial differential equation are nonnegative to detect the free boundary value and this method is very fast, also get high accuracy even when Δx is large ($\Delta x \geq 0.1$). Compact finite difference method two needs to solve an algebraic nonlinear equation (refer to [13]) at every time step, but this method can obtain second order accuracy, so it is comparable with Crank Nicholson projected SOR method. Compact finite difference method three refines the free boundary value based on compact finite difference method two by a method developed by Barone-Adesi and Lugano (1991), then use the compact finite difference method on it, and this method is highly accurate and is easily parallized.

3.1. Compact Finite Difference Method One

We develop compact finite difference method one for American option pricing problem based on the fact $u(x, \tau)$ is always postive. If $u(x, \tau)$ is negative, it is an indication that this option should be exercised. We use this fact to implicitly detect free boundary values during the implementation.

Combine the implicit identification of the free boundary values and our compact finite difference method, we use the following procedure to solve the American option pricing problem:

(i) Set initial conditions: $V(0)$ by Equ. (3.2).

(ii) Suppose we know the values of $V(\tau^m)$ at time step m, and want to compute for $V(\tau^{m+1})$ at time step $m+1$. We solve the system of linear equations $AV''(\tau^m) = b(\tau^m)$ and $b(\tau^m) = MV(\tau^m) + H(\tau^m)$ for $V''(\tau^m)$ by compact finite difference method, say $V''(\tau^m) = \Phi(V^m)$ at each time step. And the vector $H(\tau^m)$ is known since the boundary conditions Equ. (3.4) and (3.5) are given. Then the matrix form of Equ. (3.1) can be rewritten as:

$$V_\tau = \Phi(V^m) + G(\tau^m). \tag{3.6}$$

Again we get an ordinary differential equation with the initial condition V^m, solve the problem Equ. (3.6) to obtain V^{m+1} by Runge-Kutta method with fourth order of accuracy.

(iii) Detect the location of free boundary value: record the last location of the nonpositive solution at time step $m+1$, say at the ith point v_i^{m+1} such that $v_i^{m+1} \leq 0$. Because of increasing property of function $u(x, \tau^{m+1})$ for the variable x, we know $v_k^{m+1} > 0, k > i$. Reset the values $v_k^{m+1} = 0, k = 1, 2, \ldots, i$. Then save the free boundary value for this time step $m+1$, if $v_i^{m+1} = 0$, then just save x_i as the free boundary value; otherwise must hold $v_i^{m+1} < 0$, and $v_{i+1}^{m+1} > 0$, we use the zero point of the unique linear equation through points (x_i, v_i^{m+1}) and (x_{i+1}, v_{i+1}^{m+1}) to approximate free boundary value.

(iv) Repeat steps 2 and 3 until $\tau^{m+1} = \frac{1}{2}\sigma^2 T$; then change the values of $V(\tau^{m+1})$ back to option prices.

From the algorithm above, we see compact finite difference method one is to use an implicit condition to detect the free boundary value and this method is very fast, also get high accuracy when $\Delta x \geq 0.1$ by our numerical experiments. For the free boundary values we get by this method are not accurate enough, so that the accuracy of option prices is low, especially the values near to the free boundary values, furthermore, compact finite difference method one does not converge.

3.2. Compact Finite Difference Method Two

From Sect. 3.1, we see if we want to improve the accuracy of option prices, we need to know the free boundary values more accurately. To meet this end, we use a method called explicit front tracking method in [13]. This method decides free boundary value by solving the follow nonlinear equation at every time step:

$$\Phi(p,\tau) = u_{N^m} + \tfrac{1}{2}(ph)^2 g(D^- + (N^m + p)h, \tau) = 0, \tag{3.7}$$

where D^- and D^+ are the lower bound and upper bound of the truncation interval respectively, free boundary value $X^*(\tau^{m+1}) = D^- + (N^m + p)h$ and $N^{m+1} = \text{floor}(D^+ - X^*(\tau^{m+1}))$. This nonlinear equation has second order accuracy. The algorithm for compact finite difference method two can be obtained easily by modifying a little on that for compact finite difference method one in Sect. 3.1.

3.3. Compact Finite Difference Method Three

The free boundary values obtained from method two are still not accurate enough. So we need to find a way to get optimal exercise values more accurately.

Barone-Adesi and Lugano (1991) proposed a method to get the accurate free boundary by solving the system of the following equations:

$$A = -p + E - S^*, \tag{3.8}$$

$$\gamma = -N(d)S^*/A, \tag{3.9}$$

$$\tfrac{1}{2}\sigma^2\gamma(\gamma-1) - r - (r-D)\gamma - F = 0, \tag{3.10}$$

with $d = (\ln(S/E) + (r - D + \frac{1}{2}\sigma^2)\tau)/(\sigma\sqrt{\tau})$, $F = \frac{\partial p}{\partial t}/A$, and $p(S,t)$ is the European put option price with the same parameters with American put. After obtaining the values of A, S^*, and γ from the above, then compute option price by

$$P(S,t) = A(t)(S/S^*)^\gamma, \text{ for } S \geq S^*. \tag{3.11}$$

Using Equs. (3.8)–(3.10) can yield accurate free boundary values, but the option price obtained by formula Equ. (3.11) is not very accurate. We combine Equs. (3.8)–(3.10) and compact finite difference method together, to get a new accurate method for American option pricing. The algorithm for compact method three is almost same as that of compact method two, except that we compute the free boundary values by Equs. (3.8)–(3.10), instead of Equ. (3.7) in the algorithm.

Note that the nonlinear Equ. (3.7) depends on values of option price at the last time step, while Equs. (3.8)–(3.10) are independent of this, which means that we can use parallel computing in compact finite difference method three to save time.

4. Numerical Stability

As we have shown in Sect. 2.3, employing compact finite difference schemes on heat equation yields:

$$V_\tau(\tau) = A^{-1}MV(\tau) + A^{-1}H(\tau) + G(\tau), \tag{4.1}$$

for the stability issue, we can only need to analyze the following linear homogeneous matrix equation of Equ. (4.1) without loss of generality:

$$V_\tau(\tau) = A^{-1}MV(\tau), \tag{4.2}$$

and the exact solutions for Equ. (4.2) are:

$$V(\tau) = Ce^{A^{-1}M\tau}, \tag{4.3}$$

where C is a constant. The stability of the numerical scheme depends on the properties of matrix $A^{-1}M$. If all the real parts of eigenvalues of matrix $A^{-1}M$ are negative, then we know compact finite difference schemes for heat equations and American option pricing problems are unconditionally stable. In fact it is shown from our numerical experiments that eigenvalues of matrix $A^{-1}M$ are all negative, in Fig. 1 we plot the eigenvalues for matrix $A^{-1}M$ with $N = 20, 50, 100, 200$ respectively.

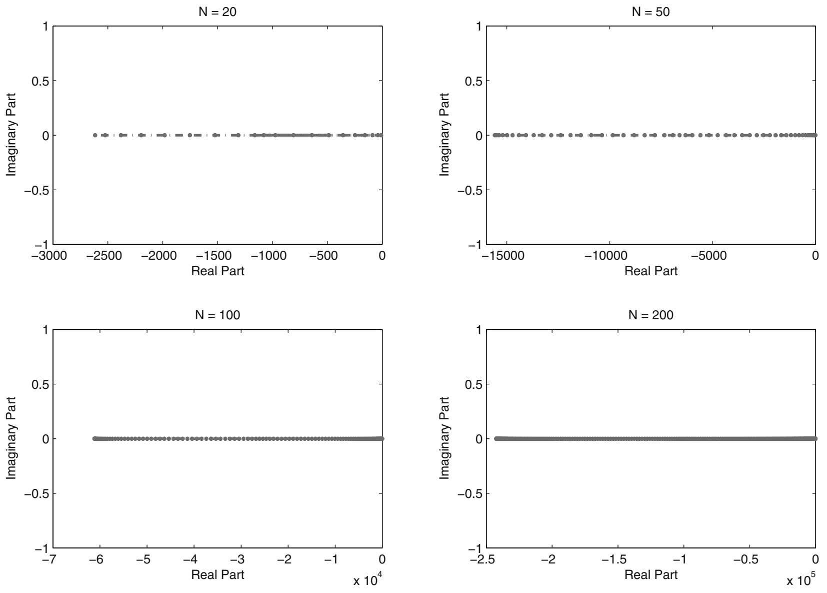


FIGURE 1. Eigenvalues for compact matrixes with varying N

5. Computational Results

In this section, we give numerical results by compact finite difference methods. Because we observe that the ode we obtain after we use compact finite difference method for American option pricing problem is stiff when τ is small, we implement our algorithms using the Matlab's powerful ode15s solver.

To compare with our results, we choose the option values obtained by trinomial method with $n = 20,000$ as the benchmark since we know that this method is convergent. We know that Crank Nicholson projected SOR method has the second order accuracy and is convergent. We determine the accuracy of our compact finite difference method by comparing Crank Nicholson projected SOR method and compact finite difference methods.

To measure the errors of our compact finite difference methods, we use root mean squared (RMS) relative error. The RMS error is defined by

$$RMS = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(\frac{\tilde{P_k} - P_k}{P_k})^2},$$

and $\tilde{P_k}$ is the approximation option prices we obtain by compact finite difference methods and P_k is the option price which we think is "true" (we use the values obtained by a 20,000 time step trinomial method).

We set the following values for the parameters of American options:

E	r	σ	D	k_1	k_2	D^-	D^+
100	.04	.3	.02	.8888889	.4444444	- 2	+ 2

We list the option prices by our compact finite difference methods in the following tables:

TABLE 1. When $\triangle x = 0.2$

		$\triangle x = 0.2$		$\triangle \tau = 0.0001$		
x	Stock Price S	Crank Nicholson	Compact method 1	Compact method 2	Compact method 3	True values
-0.2	83.9457	19.0604	19.13811	19.26822	19.51910	19.496910
0	102.5315	9.12163	9.35873	9.474996	9.60421	9.843537
0.2	125.2323	3.39863	3.47309	3.46295	3.50414	3.833369
RMS	–	.081	.065	.042	.013	–

TABLE 2. When $\triangle x = 0.1$

		$\triangle x = 0.1$		$\triangle \tau = 0.0001$		
x	Stock Price S	Crank Nicholson	Compact method 1	Compact method 2	Compact method 3	True values
-0.3	75.9572	25.2462	25.13063	25.24257	25.26206	25.329862
-0.2	83.9457	19.3749	19.30428	19.39379	19.42878	19.496910
-0.1	92.7743	14.1062	14.07499	14.14775	14.19061	14.262648
0	102.5315	9.67344	9.67773	9.72538	9.76840	9.843537
0.1	113.3148	6.21132	6.23089	6.25590	6.29326	6.365579
0.2	125.2323	3.7184	3.73115	3.74193	3.77034	3.833369
0.3	138.4031	2.07034	2.06590	2.06956	2.08854	2.137839
RMS	–	.030	.063	.029	.021	–

Table 3. When $\triangle x = 0.05$

		$\triangle x = 0.05$		$\triangle \tau = 0.0001$		
x	Stock Price S	Crank Nicholson	Compact method 1	Compact method 2	Compact method 3	True values
-0.3	75.9572	25.3092	25.1291	25.30609	25.31373	25.329862
-0.2	83.9457	19.4659	19.3296	19.46919	19.47754	19.496910
-0.1	92.7743	14.2229	14.1408	14.23126	14.24079	14.262648
0	102.5315	9.80054	9.75972	9.81129	9.82128	9.843537
0.1	113.3148	6.32658	6.30826	6.33576	6.34505	6.365579
0.2	125.2323	3.80388	3.79488	3.80841	3.81609	3.833369
0.3	138.4031	2.12009	2.11319	2.11915	2.12480	2.137839
RMS	–	.0074	.062	.0079	.0053	–

Table 4. When $\triangle x = 0.02$

		$\triangle x = 0.02$		$\triangle \tau = 0.0001$		
x	Stock Price S	Crank Nicholson	Compact method 1	Compact method 2	Compact method 3	True values
-0.3	75.9572	25.3265	25.10042	25.3257	25.32739	25.329862
-0.2	83.9457	19.4918	19.34597	19.49193	19.49383	19.496910
-0.1	92.7743	14.2561	14.16375	14.25707	14.25914	14.262648
0	102.5315	9.8365	9.78167	9.83789	9.84000	9.843537
0.1	113.3148	6.35927	6.32881	6.36044	6.36241	6.365579
0.2	125.2323	3.82849	3.81244	3.82898	3.83064	3.833369
0.3	138.4031	2.13484	2.12653	2.13451	2.13578	2.137839
RMS	–	.0012	.070	.0014	8.3e-004	–

Table 5. When $\triangle x = 0.01$

		$\triangle x = 0.01$		$\triangle \tau = 0.0001$		
x	Stock Price S	Crank Nicholson	Compact method 1	Compact method 2	Compact method 3	True values
-0.3	75.9572	25.329	25.1119	25.32869	25.32673	25.329862
-0.2	83.9457	19.4956	19.348	19.49545	19.49362	19.496910
-0.1	92.7743	14.2609	14.1665	14.26101	14.25934	14.262648
0	102.5315	9.84172	9.78475	9.84192	9.84046	9.843537
0.1	113.3148	6.36402	6.33174	6.36420	6.36296	6.365579
0.2	125.2323	3.83209	3.81494	3.83215	3.83116	3.833369
0.3	138.4031	2.13702	2.12844	2.13692	2.13619	2.137839
RMS	–	.00031	.066	.00039	.001	–

From the above results, we see that the compact finite difference method one also gets high accuracy when Δx is not too small ($\Delta x \geq 0.1$). While Δx is too small, like $\Delta x = 0.01$, the result does not obtain high accuracy since this method does not converge, which can be explained by the poor free boundary values obtained from implicit method. The compact difference method two can obtain second order accuracy, so it is comparable with Crank Nicholson projected SOR method. The compact difference method three is more accurate than Crank Nicholson projected SOR and compact method two when $\Delta x \geq 0.02$. For the free boundary values we use for the method three only obtain $O(1/5000)$, so when $\Delta x = 0.01$, the results of compact method three are not more accurate than them.

We can see that our compact finite difference methods converge rapidly, but the error order of compact finite difference method does not obtain $O(h^4)$. The reason is that the free boundary values obtained by our three compact finite difference methods are not accurate enough. To this point, we can compare the free boundary values by compact finite difference method one, compact finite difference method two, the integral method of Kim [11], and the analytic approximations of Barone-Adesi and Elliott [2] (used in compact finite difference method three). And the "true" values for free boundary are based on analytical approximations method when time steps $n = 5,000$, other methods are based on $n = 50$ in Fig. 2.

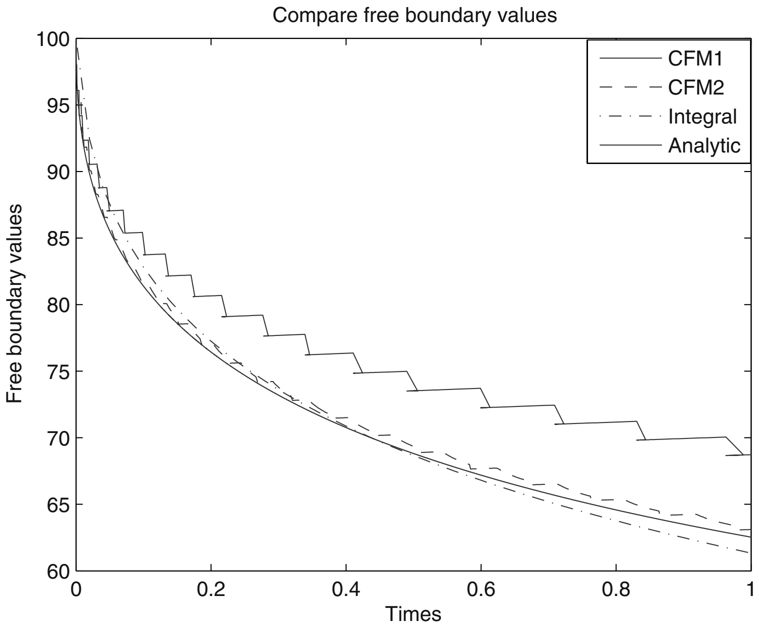


FIGURE 2. Free boundary values for American Put Option

From Fig. 2, we can see that the accuracy of compact finite difference method one for free boundary value is poor; the compact finite difference method two can obtain second order for free boundary values and this method converges from our experiments; and the compact finite difference method three can obtain higher order accuracy than compact method two since this method use analytic approximations of Barone-Adesi and Elliott [2] to find the location of free boundary conditions with smaller space step Δx.

6. Conclusions

By showing how to adjust compact finite difference methods for heat equation and American option pricing problems, we have shown that compact finite difference method is very flexible and can obtain high accuracy under some conditions.

It seems that the accuracy and speed of our compact finite difference methods for American option pricing problem depend heavily on the method we use to obtain the free boundary values. The compact finite difference method one can rapidly get high accuracy even when Δx is not too small ($\Delta x \geq 0.1$). While Δx is too small, like $\Delta x = 0.01$, the result fails to obtain high accuracy. The compact finite difference method two can obtain second order accuracy, so it is comparable with Crank Nicholson projected SOR method. The compact finite difference method three is more accurate than Crank Nicholson projected SOR method, but this method pays more times on it.

Acknowledgment

We thank the two anonymous referees for their very useful comments and discussions on this paper.

Appendix: Maple Codes to Generate Fourth-Order Compact Finite Difference Formulae for Second Derivatives Symbolically

```
restart;
f1 := (m[-1]*diff(u(x-h),x,x)+ m[0]*diff(u(x),x,x)
      + m[1]*diff(u(x+h),x,x)) - (a[-1]*u(x - h)
      + a[0]*u(x) + a[1]*u(x+h))/h^2:
f1 := convert(series(f1, h, 7),polynom):
for i from 1 to 7 do
    if i = 1 then
        eqn[i] := simplify(coeff(f1,h,i-3)/u(x));
    else   eqn[i] := simplify(coeff(f1,h,i-3)/(D@@(i-1))(u)(x));
    end if:
end do:
sols := solve( {eqn[1],eqn[2],eqn[3],eqn[4],eqn[5],eqn[6]},
             {m[-1],m[0],m[1],a[-1],a[0],a[1]} );
formula1 := (m[-1]*diff(u(x-h),x,x)+ m[0]*diff(u(x),x,x)
             + m[1]*diff(u(x+h),x,x))
             - (a[-1]*u(x - h) + a[0]*u(x) + a[1]*u(x+h))/h^2:
formula1 := subs(sols,formula1);
```

References

[1] G. Barone-Adesi and R. Whaley, Efficient analytic approximation of American option values, Journal of Finance, pp. 301-320, June 1987.

[2] G. Barone-Adesi and R. Elliott, Approximations for the values of American options, Stochastic Analysis and Applications, Vol. 9(2), pp. 115-131, 1991.

[3] G. Barone-Adesi and U. Lugano, The Saga of the American Put, 2003.

[4] M. Ahmed, An exploration of compact finite difference methods for the numerical Solution of PDE, Ph.D. thesis, the University of Western Ontario, 1997.

[5] M.H. Carpenter, D. Gottlieb and S. Abarbanel, The stability of numerical boundary treatments for compact high-order finite-difference schemes, Journal of Computational Physics, Vol. 108, pp. 272–295, 1993.

[6] R.M. Corless, J. Rokicki and J. Zhao, FINDIF, a routine for generation of finite difference formulae. Share Library package 1994, and upgraded to n dimensions for "iguana", 2006.

[7] B. Düring, M. Fournié and A. Jüngel, High order compact finite difference schemes for a nonlinear Black-Scholes equation, 2005.

[8] B. Düring, M. Fournié and A. Jüngel, Convergence of a high-order compact finite difference schemes for a nonlinear Black-Scholes equation, 2005.

[9] Y. Gu, J. Shu, X. Deng and W. Zheng, A new numerical method on American option pricing. Science in China, Vol. 45(3), pp. 181–188, 2002.

[10] C.H. John, Options, Futures, and other Derivatives, 4th edition, Prentice-Hall, Inc., 1999.

[11] I.J. Kim, The analytic valuation of American options, Review of Financial Studies, Vol. 3, pp. 547–572, 1990.

[12] S.K. Lele, Compact finite difference schemes with spectral-like resolution, Journal of Computational Physics, Vol. 103, pp. 16–42, 1992.

[13] K.N. Pantazopoulos, Numerical methods and software for the pricing of American financial derivatives. Ph.D. thesis, Computer Science Dept., Purdue University, 1998.

[14] W.F. Spotz, High-order compact difference schemes for computational mechanics. Ph.D thesis, University of Texas at Austin, 1995.

[15] H. Sun, and J. Zhang, A high order compact boundary value method for solving one dimensional heat equations, Technical Report No. 333-02, University of Kentucky, 2002.

[16] P. Wilmott, J. Dewynne and S. Howison, Option Pricing: Mathematical Model and Computation, Oxford Financial Press, 1995.

Jichao Zhao and Robert M. Corless (Distinguished University Professor)
Ontario Research Centre for Computer Algebra
Department of Computer Science
University of Western Ontario
Middlesex College, London
1151 Richmond St.
Ontario, Canada N6A 5B7
e-mail: jzhao29@uwo.ca
rcorless@uwo.ca

Matt Davison
Department of Applied Mathematics
University of Western Ontario
Middlesex College, London
1151 Richmond St.
Ontario, Canada N6A 5B7
e-mail: mdavison@uwo.ca

Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 375–394
© 2007 Birkhäuser Verlag Basel/Switzerland

Symbolic Analyzer for Large Lumped and Distributed Networks

Majid A. Al-Taee, Fawzi M. Al-Naima and Bessam Z. Al-Jewad

Abstract. A symbolic linear analyzer that integrates the analysis of both lumped and distributed networks is presented in this paper. This is achieved by implementing a consistent set of notations and a unified mathematical framework. The merits of sparse symbolic matrices are utilized to develop a complete symbolic sparse toolbox that includes all basic elements of electrical circuits. A new storage scheme called Row-Indexed Semi-Symetric (RISS) sparse is proposed and implemented for symbolic network matrices. Several existing and newly developed matrix solvers are implemented and their performances are compared. These solvers are used in the proposed symbolic solver to provide a multi-path analysis root that can be optimized for a particular problem. The overall performance of the proposed symbolic analyzer is assessed practically using several application examples. The obtained results are found in agreement with the results obtained from previously reported analyses and thus confirm validity of the adopted formula approximation method. A polynomial fitting shows that the time complexity of a variable size RC network is super-quadratic with the number of network sections. A substantial reduction in the analysis time of a large ladder RC network is demonstrated with the implementation of macromodeling techniques.

Keywords. Symbolic analysis, symbolic matrix storage, lumped and distributed network analysis, optimal computer solution.

1. Introduction

Computer-aided simulation of circuits is a mature field, as evidenced by the wide usage of circuit simulation programs. The great majority of the currently available programs belong to the numeric category in the sense that their outputs are numbers. Symbolic network analysis, on the other hand, is aimed at producing outputs

This work was completed with the support of Computer Engineering Department at Al-Nahrain University, Iraq.

as expressions that contain variables and numbers. Thus symbolic analysis can be viewed as an essential complement to numerical simulation in the design and evaluation of analog circuits. It provides insight into circuit behavior that numerical analysis does not. Numerous symbolic-analysis tools have been developed for analog circuits [1–6]. The increasing computation power and use of more efficient algorithms enable symbolic analysis tools to handle larger circuits than before. The circuit size that can be handled by these tools however is still much smaller than those handled by numerical simulators. The primary difficulty is the exponential growth of product terms in a symbolic network function with respect to the circuit size. This long-standing difficulty is only partially overcome by various symbolic approximation and hierarchical decomposition approaches [7, 8].

Symbolic approximation can be performed before, during or after the generation of symbolic terms [8–10]. The approximated expressions however only have sufficient accuracy over some frequency ranges. Moreover, approximation often loses some information that is crucial for circuit optimization. Hierarchical decomposition approaches are based on the sequence-of-expressions concept to obtain the network transfer function [10, 11]. The decomposition process initially eliminates variables one at a time in each sub-circuit. The obtained results from sub-circuits analyses are then combined to form the upper-level circuit equations. In this approach, the transfer function is trivially computed from the resulting equations and the intermediate results are not compact enough. The number of expressions therefore grows rapidly with respect to the circuit size. Consequently, the CPU operation count and memory storage requirement are also increased. A graph-based hierarchical method for the generation of exact symbolic network functions is reported in [4, 5, 10]. This method employs determinant decision diagram that is a signed rooted directed acyclic graph with two terminal vertices, namely the 0-terminal vertex and the 1-terminal vertex. The main advantage of this method lies in the sharing of the sub-expressions and the zero suppression in the vertices. The number of CPU operations however also increases rapidly as the number of nonzero symbolic coefficients increases.

Algorithmic solutions for symbolic analysis evolved from two basic needs: analysis of very large interconnected systems with many symbolic variables and simplification of the generated symbolic expressions to facilitate human interpretation. These two distinct and typically non-compatible needs resulted in two different paths for algorithm development. However, the recently developed techniques such as the symbolic formula approximation, circuit shifting, and the hierarchical decomposition have offered the possibility of unifying the paths of algorithm development.

In this paper, a symbolic analyzer that combines the analysis of lumped and distributed networks in one program is described. The development process of this analyzer is based on a unified treatment of symbolic methods using a consistent set of notations. The merits of sparse symbolic matrices are utilized to develop a complete symbolic sparse toolbox that includes all basic elements of electrical circuits, namely: the independent sources, passive elements, controlled current and voltage sources, nullators, norators and fixators. Several other existing solvers that are dedicated to special types of problems are also implemented and evaluated. These solvers are used to provide a multi-path analysis root that can be optimized for a particular problem. The performance of the proposed symbolic analyzer are assessed using two simulation examples, the 741 op-amp and ladder RC networks of up to 1000 identical sections.

The rest of the paper is organized as follows: Section 2 describes the main procedures involved in the design of the symbolic analyzer. Section 3 describes a new efficient storage scheme called Row-Indexed Semi-Symmetric Sparse (RISS) for symbolic network matrices. Performance comparisons of some existing matrix solvers that are used in the developed solver are also presented in this section. Section 4 summarizes the role of approximation and shifting algorithms and presents a practical procedure for implementing these algorithms in the proposed symbolic analyzer. Section 5 presents simulation results and performance evaluation for the developed symbolic solver. Section 6 concludes the paper.

2. Structure of the Symbolic Analyzer

A complete symbolic sparse toolbox that offers library creation property is developed in the MAPLE environment [12]. The script codes of the developed programs are compiled as standalone functions compatible with MATLAB. The structure of this analyzer is composed of four main procedures; an input, data assembly and modification, symbolic linear solver, and an output procedure. The main functions performed by these procedures are shown in Fig. 1 and are described briefly as follows.

2.1. Input

This procedure accepts a script file containing a description of the circuit topology in a SPICE-like language. This file contains the netlist, description of the circuit under analysis, and other information related to th analysis process such as the models of semiconductor devices. Typical circuit elements include resistors, capacitors, inductors, independent sources, semiconductor devices, and functional blocks, such as op-amps. Example of an active-compensated amplifier circuit is shown in Fig. 2 and the corresponding input file is shown in Fig. 3. The contents of this file can be described as follows; the upper part describes the circuit topology and includes some numbers that guide the simplification process. The middle part specifies a four-level model for PMOS and NMOS transistors. Finally, the lower part specifies the required analysis and control parameters. The first line of the latter part orders the computation of transfer function for voltages at nodes 1 and 2. The second line orders the computation of a simplified formula and its associated control parameters. Next line requests pole-zero extraction. The last line of this part orders the realization of parametric analysis based on varying the capacitor C_{pz}.

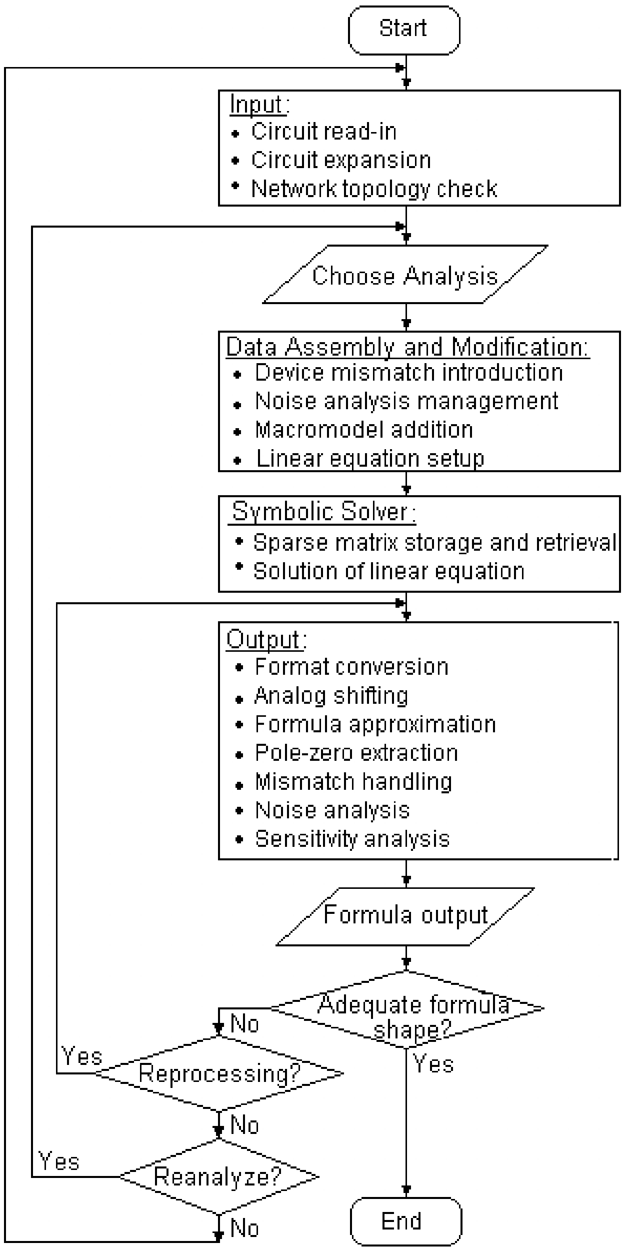


FIGURE 1. Simplified flowchart of the proposed symbolic analyzer

2.2. Data Assembly and Modification

This procedure links the input data file with the different computational procedures after formulating the system matrix. It prepares and checks all data associated with the various model and functional block representations and modifies the

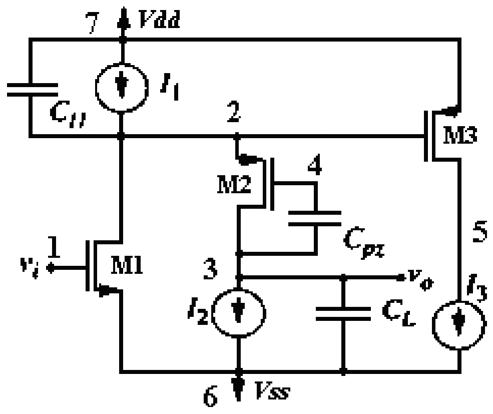
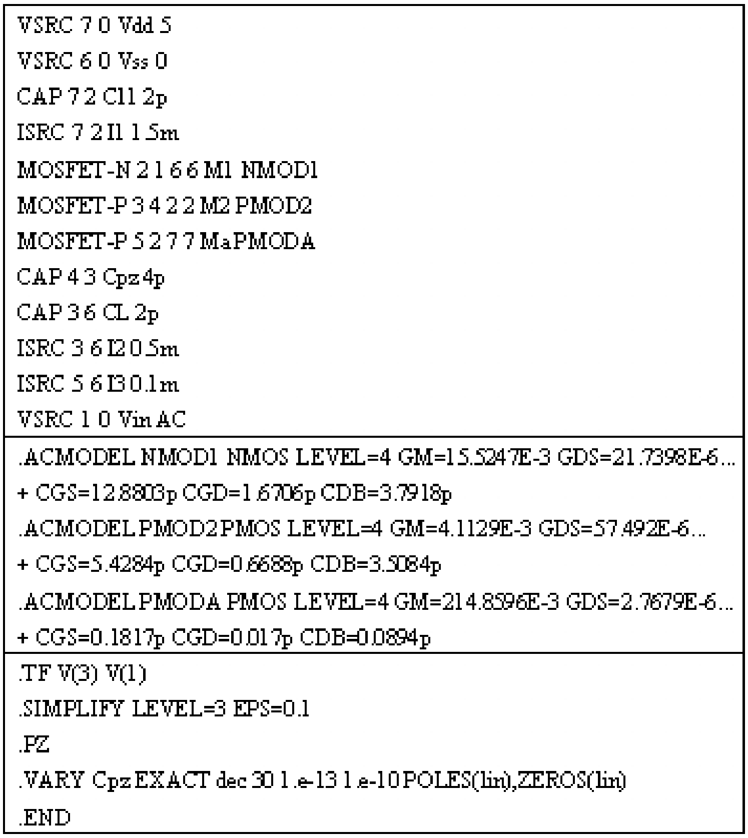


FIGURE 2. Circuit of an active-compensated amplifier

system data by adding device equivalent circuits. In addition, this procedure also performs the necessary data modifications for noise analysis. The analyzer also includes strategies to simplify handling of matched devices by including dedicated procedures. These procedures handle mismatches (due to variations in fabrication processes) that play a dominant role in some second-order small-signal circuit characteristics. The total mismatching effect is quantified in the output function in terms of delta variables. Each of these variables can be assigned some probability function and handled as a random variable to perform a local-area tolerance analysis. When all the devices are assigned mismatching values from nominal design values, a general tolerance analysis can be performed.

2.3. Symbolic Linear Solver

The symbolic linear solver includes several solution procedures for a set of symbolic linear equations describing the network as well as the the procedures of sparse storage and retrieval. The performance and output types of this part determine the overall performance of the symbolic analyzer. The network solution is usually obtained in a specific format that should facilitate the interpretation and manipulation of the output terms. Simultaneously, the output format should also minimize the storage requirements and the computational complexity. When analyzing large circuits however, the storage requirements become the dominant concern. In this case, the use of Sequence of Expressions (SOE) format of the output becomes an unavoidable choice despite its interpretation and approximation difficulty [5]. This method is computationally efficient and capable of making the time performance of the symbolic analyzer comparable to that of the numerical analyzer. This is justified by practical measurements presented in Sect. 5.

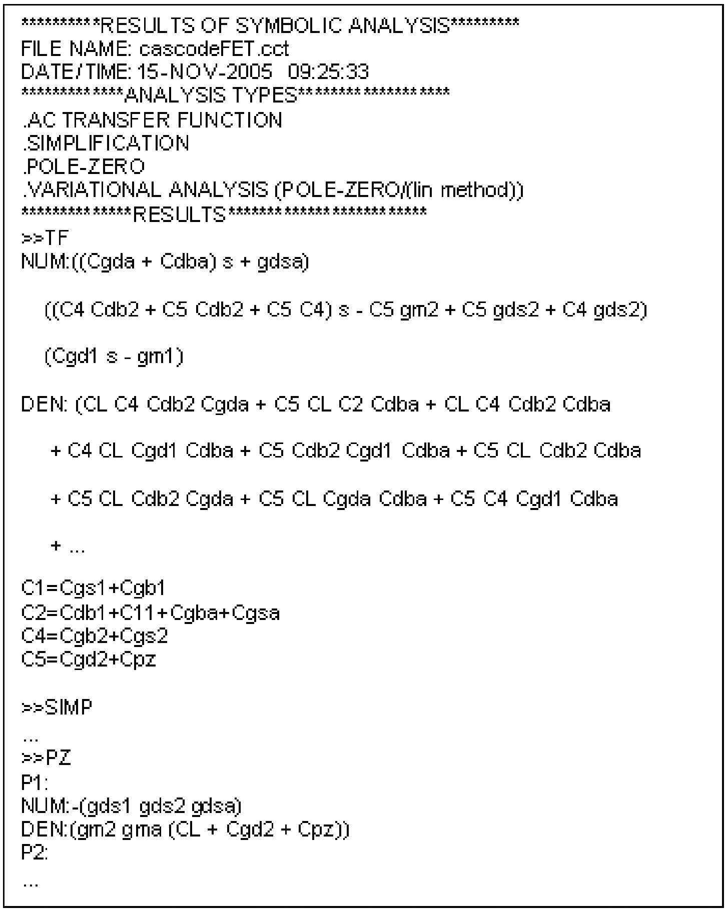


```
VSRC 7 0 Vdd 5
VSRC 6 0 Vss 0
CAP 7 2 Cl1 2p
ISRC 7 2 I1 1.5m
MOSFET-N 2 1 6 6 M1 NMOD1
MOSFET-P 3 4 2 2 M2 PMOD2
MOSFET-P 5 2 7 7 Ma PMODA
CAP 4 3 Cpz 4p
CAP 3 6 CL 2p
ISRC 3 6 I2 0.5m
ISRC 5 6 I3 0.1m
VSRC 1 0 Vin AC
.ACMODEL NMOD1 NMOS LEVEL=4 GM=15.5247E-3 GDS=21.7398E-6...
+ CGS=12.8803p CGD=1.6706p CDB=3.7918p
.ACMODEL PMOD2 PMOS LEVEL=4 GM=4.1129E-3 GDS=57.492E-6...
+ CGS=5.4284p CGD=0.6688p CDB=3.5084p
.ACMODEL PMODA PMOS LEVEL=4 GM=214.8596E-3 GDS=2.7679E-6...
+ CGS=0.1817p CGD=0.017p CDB=0.0894p
.TF V(3) V(1)
.SIMPLIFY LEVEL=3 EPS=0.1
.PZ
.VARY Cpz EXACT dec 30 1.e-13 1.e-10 POLES(lin),ZEROS(lin)
.END
```

FIGURE 3. Sample input file

2.4. Output

This procedure uses the network solution to obtain the required outputs that express system functions as quotients of polynomials. The developed symbolic analyzer contains a format changing procedure to present the output according to the user specification. It also contains a numerical substitution procedure to substitute for some or all the variables. Other optional outputs include symbolic poles and zeros, sensitivity analysis, and term approximation. A sample output file is shown in Fig. 4. The information provided in this file includes the title of analysis, file name, some remarks and a list for the required analyses. Next, the obtained results are presented in the required format for each analysis, as illustrated.



```
**********RESULTS OF SYMBOLIC ANALYSIS*********
FILE NAME: cascodeFET.cct
DATE/TIME: 15-NOV-2005  09:25:33
*****************ANALYSIS TYPES*******************
.AC TRANSFER FUNCTION
.SIMPLIFICATION
.POLE-ZERO
.VARIATIONAL ANALYSIS (POLE-ZERO/(lin method))
*******************RESULTS*************************
>>TF
NUM:((Cgda + Cdba) s + gdsa)

   ((C4 Cdb2 + C5 Cdb2 + C5 C4) s - C5 gm2 + C5 gds2 + C4 gds2)

   (Cgd1 s - gm1)

DEN: (CL C4 Cdb2 Cgda + C5 CL C2 Cdba + CL C4 Cdb2 Cdba

    + C4 CL Cgd1 Cdba + C5 Cdb2 Cgd1 Cdba + C5 CL Cdb2 Cdba

    + C5 CL Cdb2 Cgda + C5 CL Cgda Cdba + C5 C4 Cgd1 Cdba

    + ...

C1=Cgs1+Cgb1
C2=Cdb1+C11+Cgba+Cgsa
C4=Cgb2+Cgs2
C5=Cgd2+Cpz

>>SIMP
...
>>PZ
P1:
NUM:-(gds1 gds2 gdsa)
DEN:(gm2 gma (CL + Cgd2 + Cpz))
P2:
...
```

Figure 4. Sample output file

3. Symbolic Matrix Operations

Three matrix operations related to circuit analysis can be identified: matrix solution of a linear system, inversion, and determinants. All these matrix operations can be used in the symbolic context of circuit analysis, and the choice between them is merely based on the processing time and storage requirements [13, 14]. As mentioned earlier in the introduction, some existing algorithms are implemented to provide a multi-path analysis root that can be optimized by the user for a particular problem. These algorithms involve several network solution methods, namely: the determinant-based methods, transform-based methods, Strassen's method, Shipley's method, Gaussian elimination, LU factorization, and the Kron's matrix-reduction method. Theoretical details of these algorithms are widely reported in literature [14–18] and thus such details are considered beyond the scope of this paper. Figure 5 shows a performance comparison between the adopted methods of matrix inversion. A similar comparison is shown in Fig. 6 for the different network solutions methods used in the developed symbolic analyzer. The obtained results show that the Gaussian elimination, LU factorization, and the Kron's matrix-reduction method have similar amounts of saving in the number of operations. Practically, however, the Kron's method works faster for large systems due to the minimization of access times to the mass memory. This method is also useful in finding the transfer characteristics of large networks and thus it is adopted during the formulation of the system matrix for computer implementation. The rest of this section discusses the implementation of a recently reported determinant-based method that employs determinant decision diagrams. A new storage scheme called Row-Indexed Semi-Symmetric Sparse (RISS) for symbolic network matrices is also presented in this section.

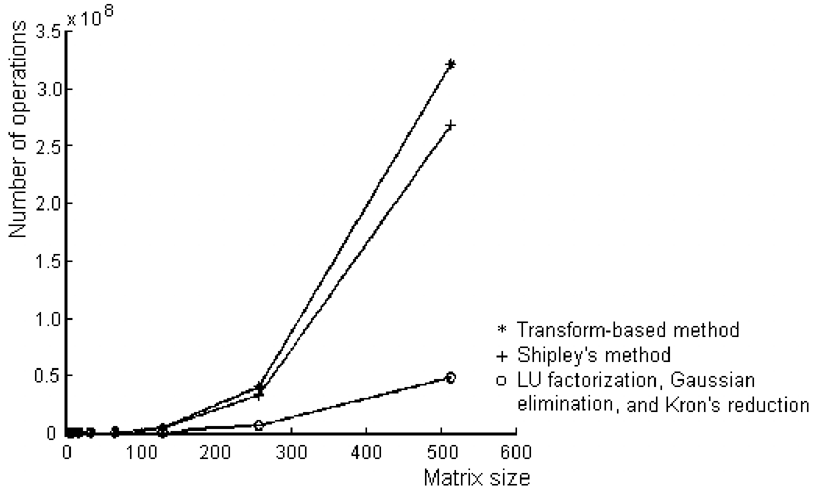


FIGURE 5. Comparison between different of matrix-inversion methods

3.1. Determinant Decision Diagrams

Symbolic network analysis results in sparse band matrices that can be efficiently solved by the Determinant Decision Diagram (DDD) method. This method is a signed rooted directed acyclic graph with two terminal vertices, namely the 0-terminal vertex and the 1-terminal vertex. Each non-terminal vertex a_i is associated with a sign, $s(a_i)$, determined by the sign rule defined in reference [19]. It has two outgoing edges, called 1-edge and 0-edge, pointing respectively to D_{a_i} and $D_{\overline{a}_i}$. A determinant decision graph having root vertex $s(a_i)$ denotes a matrix determinant D defined recursively as follows:

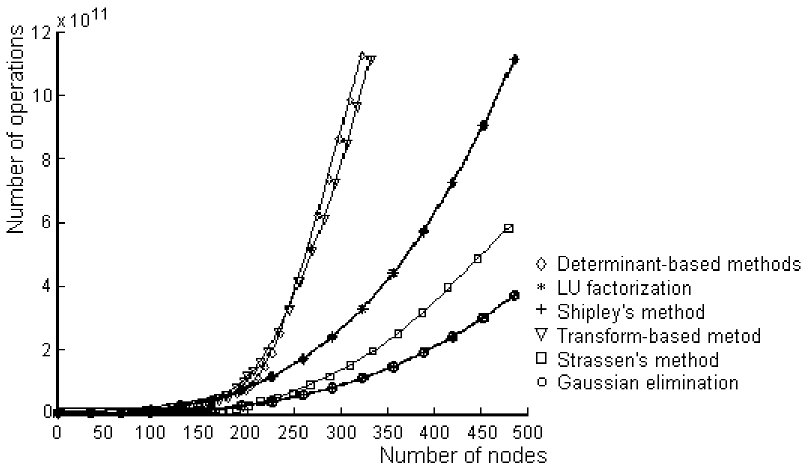


FIGURE 6. Comparison between different network-solution methods

- If a_i is the 1-terminal vertex, then $D = 1$;
- If a_i is the 0-terminal vertex, then $D = 0$;
- If a_i is a non-terminal vertex, then $D = a_i s(a_i) D_{a_i} + D_{\overline{a}_i}$,

where $s(a_i)D_{a_i}$ is the cofactor of D with respect to a_i, $D_{\overline{a}_i}$ is the minor of D with respect to a_i, and D_{a_i} is the remainder of D with respect to a_i. For example, consider the determinant of Equ. 3.1 [5]:

$$det = \begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj - adhi - aefj - bcgj + bchi. \tag{3.1}$$

The DDD for this matrix (under the vertex ordering $a > c > b > d > f > e > g > i > h > j$) is shown in Fig. 7(a). This method is a direct implementation of the Laplace expansion and it has the same time complexity. The main advantage of this method lies in the sharing of the sub-expressions and the zero suppression in the vertices. However for a heavily connected network such as that represented by the determinant of Equ. 3.2 (under the vertex ordering $a > d > g > e > b > h > c > f > i$), the efficiency of this method is reduced as illustrated in Fig. 7(b):

$$det = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei - afh + bfg - bdi + cdh - ceg. \tag{3.2}$$

The algorithm needed to automate this process can be combined with sparse matrix algorithms to find the determinant. The number of operations required for this method increases rapidly as the number of nonzero symbolic coefficients increases.

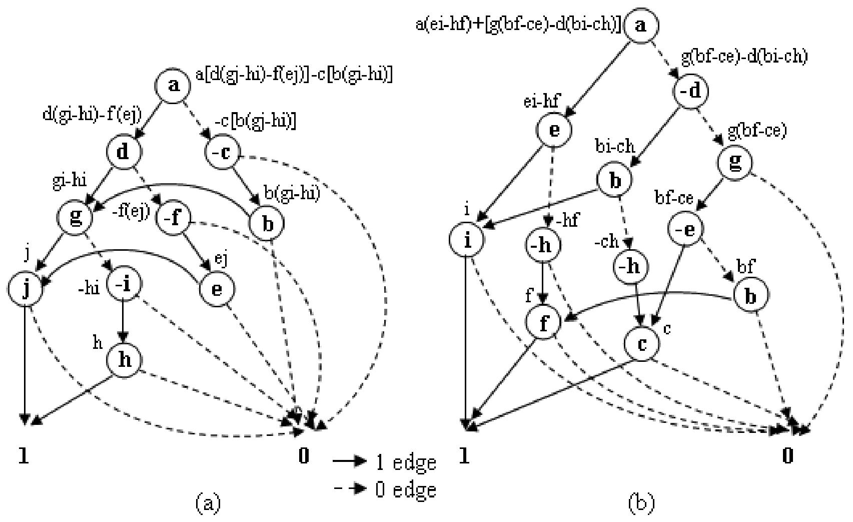


FIGURE 7. The determinant decision diagrams

3.2. Indexed Storage of Sparse Matrices

The use of sparse matrix techniques results in substantial reduction in computing cost even for relatively small problems. However, sparse matrix software development is a rather difficult task and is completely application dependent. The solution of sparse systems of equations involves two kinds of techniques: efficient storage and retrieval of the nonzero matrix entries, and efficient implementation of the logic of the linear solution algorithms to exploit storage efficiencies. As an example, consider the nodal formulation of a typical network with N ungrounded nodes and b two-terminal passive components. The admittance matrix Y contains at most $N + 2b$ nonzero entries. In a typical large network, the number of components ranges from two to four times the number of nodes [14]. The total number of nonzeros in the matrix of such networks is only a small fraction of N^2 (the total number of entries in the matrix).

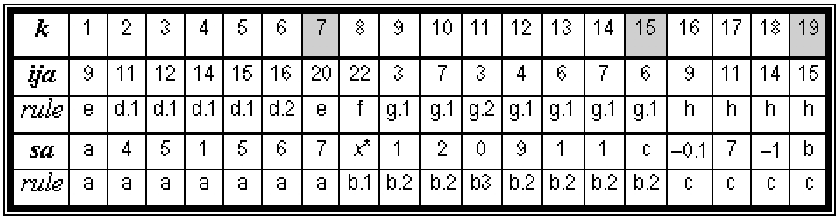
Inspired by the fact that the actual computation time is very long compared with the matrix storage, reordering and retrieval time, a new storage scheme called RISS is proposed. This scheme requires storage of at most about two times the number of nonzero matrix elements. Taking into consideration that the symbolic entries of the matrix may take a huge memory size compared to the integers representing their indices, this method takes about the same memory space needed to store only the nonzero entries for fully symbolic sparse matrices. For example, to represent a matrix (say matrix A) of dimension $N \times N$, the RISS scheme sets up two one-dimensional arrays, call them *sa* and *ija* stores. The following steps summarize the storage rules of the RISS scheme:

- a The first N locations of sa stores the A's diagonal matrix elements, in order. Note that diagonal elements are stored even if they are zeros; this is a slight storage inefficiency since diagonal elements are nonzero in most network applications.
- b This rule can be divided into three sub-rules, as follows:
  - b.1 Location $N + 1$ of sa is not used.
  - b.2 Entries at locations $\geq N + 2$ contain A's non-diagonal values of the upper triangle ordered by rows, and ordered by columns within each row.
  - b.3 If a column in the lower triangle contains an element that is not found in the associated row location of the upper triangle, a zero is inserted in sa in that place. This is also a slight storage inefficiency since in network applications the matrices are often structurally symmetric. This property however is formulation dependent and some reordering of the matrix may be needed.
- c After scanning all the rows of the upper triangle, the columns of the lower triangle are scanned. Any element that has the same value at the same corresponding position in the upper triangle is not stored. Should an element be found to have some discrepancy from its associated upper triangle image then the latter is subtracted from the element and the discrepancy is stored in sa. The elements are scanned by columns, and by rows, within each column.
- d This rule can be divided into two sub-rules, as follows:
  - d.1 Each of the first $N - 1$ locations of ija stores the index of the array sa that contains the first non-diagonal element of the corresponding row of the upper triangle of the matrix.
  - d.2 If there are no non-diagonal elements for that row, it is one greater than the index in sa of the most recently stored element of a previous row.
- e Location 1 of ija is always equal to $N + 2$. (It can be read to determine n.) While location n stores the (length of sa)+1.
- f Location $N+1$ of ija is double the index of sa of the last element of the last row in the upper triangle minus $(N + 1)$. It can be read to determine the number of the elements that are considered nonzero in the matrix.
- g This rule can be divided into two sub-rules, as follows:
  - g.1 Entries in ija at locations $\geq N + 2$ contain the *column* numbers of the corresponding elements in sa for all elements of the upper triangle.
  - g.2 If elements of sa are zero, then entries in ija at locations $\geq N + 2$ contain the *row* numbers of the corresponding elements in sa for all elements of the lower triangle.
- h The remaining entries of ija contain the indices of the array sa that contain the elements to which the discrepancies should be added to construct the lower triangle elements.

Implementation of the above rules can be explained with reference to the following symbolic matrix example:

$$\begin{bmatrix} a & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0.9 & 7 & 5 & 9 & 0 & 1 & 0 \\ 0 & 0 & 9 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 5 & c & 0 \\ 0 & 0 & 1 & 0 & c+b & 6 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 7 \end{bmatrix}.$$

In the RISS compact storage, this matrix is represented by the two arrays of length 19 shown in Fig. 8. Notice that, according to the storage rules, the value of N (namely 7) is ija[1]-2, and the length of each array is ija[ija[1] - 2] - 1, namely 19. To clarify the example the following variables with their associated values will be assumed: F = ija[1] - 2 = 7, L = ija[ija[1] - 2] - 1 = 19, and M = (ija[ija[1] - 1] + ija[1] - 1)/2 = 15. The diagonal element in row i is sa[i] and the upper-triangle elements in that row are in sa[k] where k ranges from ija[i] to ija[i+1]-1 except for the last row where it loops until M as long as the upper limit of the loop is greater than or equal to the lower limit. The required column values are stored in ija[k]. The lower triangle elements of column i are stored in sa[k] (where the limits of the k loop are the same as those given above) added to it the values of sa[j], where j loops from $M+1$ to L if there is a value ija[j] that matches k. Such an arrangement is well suited for Crouts algorithm [13, 14]. When the factorization (or in the same sense the elimination) is based on a row-by-row analysis, a complete row has to be generated.



k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ija	9	11	12	14	15	16	20	22	3	7	3	4	6	7	6	9	11	14	15
rule	e	d.1	d.1	d.1	d.1	d.2	e	f	g.1	g.1	g.2	g.1	g.1	g.1	g.1	h	h	h	h
sa	a	4	5	1	5	6	7	x*	1	2	0	9	1	1	c	–0.1	7	–1	b
rule	a	a	a	a	a	a	a	b.1	b.2	b.2	b3	b.2	b.2	b.2	b.2	c	c	c	c

**x is an arbitrary value*

Figure 8. Example of the RISS compact storage scheme

An algorithm that converts a matrix from full storage mode into RISS storage mode is developed and implemented in the proposed symbolic analyzer. The principle application of sparse storage mode is for matrices whose full storage mode will not fit into the machine at all; then one has to transform them directly into sparse format. This algorithm is therefore serves as an algorithmic definition of the storage scheme. It is also useful for subscale testing of large problems and for the case where execution time, rather than storage, furnishes the impetus to sparse storage. Retrieval of the full matrix from RISS mode can simply be performed by reversing the algorithm.

4. Approximation and sifting Algorithms

Approximating an expression means pruning insignificant terms based on their magnitudes. Mathematically, it is trading off accuracy (error) against simplicity (complexity). The approximation process can be performed before or after solving the system matrix. However, the approximation after solving the system matrix can be very expensive for large circuits [2]. In order to limit the effects of exponential time and storage complexity, the approximation process should therefore be performed before solving the system matrix. In the developed symbolic solver, the formula approximation is achieved by implementing a number of analog sifting algorithms, namely [2, 7, 14]: the device-parameter elimination, successive reduction, and the cancellation algorithm. These algorithms are briefly described as follows.

4.1. Device-Parameter Elimination

This process aims at eliminating device parameters in the system matrix when the elimination does not cause significant numerical error. The value of the system matrix determinant is used as an error evaluation criterion. For example, a determinant with $N \times N$ dimension can be expanded with respect to row a and column j as follows:

$$|M| = \sum_{j=1}^{N} m_{a,j}(-1)^{a+j}|M_{a,j}|, \tag{4.1}$$

where $m_{a,j}$ is the element in the a^{th} row and j^{th} column of M. $|M_{a,j}|$ is the determinant of the sub-matrix of M formed by deleting the a^{th} row and j^{th} column. Taking the derivative of the determinant with respect to a certain matrix element (say, the element in the the a^{th} row and b^{th} column) yields

$$\frac{d|M|}{dm_{a,b}} = (-1)^{a+b}|M_{a,b}|. \tag{4.2}$$

The change in the value of the determinant caused by adding a perturbation in the a^{th} row and b^{th} column is thus given by

$$\varepsilon = |\tilde{M}| - |M| = \Delta(|M|) \approx \Delta m_{a,b}\frac{d|M|}{dm_{a,b}}, \tag{4.3}$$

where $\Delta m_{a,b}$ is the perturbation at row a and column b. Substituting Equ. 4.2 in Equ. 4.3 yields

$$\varepsilon \approx \Delta m_{a,b}(-1)^{a+b}|M_{a,b}|. \tag{4.4}$$

The perturbed matrix is defined as

$$\tilde{M} = M + \Delta M, (\Delta M)_{i,j} = \left\{ \begin{array}{ll} \Delta m_{a,b} & \text{if } i = a \text{ and } j = b, \\ 0 & \text{otherwise.} \end{array} \right. \tag{4.5}$$

Thus, four, two (in the same row and column), or one of the device parameters can be removed from a cofactor if the removal causes an error in the value of the cofactor $\Delta(|M|)$ less than the desired error margin. The locations (row and column indices) of the m (one, two or four) perturbations in the matrix are represented by a_k, b_k. According to the symmetrical appearances of each device, Equ. 4.3 can be easily extended, for the case of m perturbations, as follows [2]:

$$\Delta(|M|) = \sum_{k=1}^{m} \Delta m_{a_k,b_k} (-1)^{a_k+b_k} |M_{a_k,b_k}|, \tag{4.6}$$

where m_{a_k,b_k} is the matrix element at the ${a_k}^{\text{th}}$ row and ${b_k}^{\text{th}}$ column. $\Delta m_{a_k,b_k}$ is the value of the perturbation at location (a_k, b_k) in the matrix. M_{a_k,b_k} is the sub-matrix of M formed by deleting the ${a_k}^{\text{th}}$ row and ${b_k}^{\text{th}}$ column. The key point here is that the right hand side of Equ. 4.5 is evaluated after substituting the nominal values of the device parameters, and thus the error needs numerical determinants. The procedure of device parameter elimination proceeds as follows; the whole device (all four device parameters) is initially removed from a cofactor. If the error in the cofactor value is greater than a specified error margin, the device is put back in the cofactor. This process is then repeated with the device parameters that lie either in the same row or the same column. If the cofactor error is still unacceptable, only a single parameter is removed. Finally, the approximated symbolic formula is evaluated.

The practical implementation of device parameter elimination process shows that the elimination algorithm has a quadratic time-complexity, i.e. the time required is proportional to b^2, where b is the number of device parameters. In a large network, b may range from $2N$ to $4N$ (N is the number of nodes) and the required time is therefore proportional to N^2 [14].

4.2. Successive Reduction

This algorithm overcomes the need for simplifying intermediate terms and performs as follows. For example, consider the following matrix equation:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}. \tag{4.7}$$

As mentioned earlier in Sect. 3, the Kron's reduction method is implemented successively until x_1 remains as the only variable. Then, after computing x_1, it is treated as a constant and subtracts the first column multiplied by x_1 from the right-hand side after, of course, eliminating the first row and column. Thus, the matrix equation becomes

$$\begin{bmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & & \vdots \\ a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_2 - a_{21}x_1 \\ b_3 - a_{31}x_1 \\ \vdots \\ b_n - a_{n1}x_1 \end{bmatrix}. \tag{4.8}$$

In this substitution, the value of x_1 will not be used but only its symbolic name. The matrix size is now reduced and we can proceed to compute x_2, which will be, in this case, a function of x_1. Then, treating x_2 as a constant also, the resulting matrix equation becomes

$$\begin{bmatrix} a_{33} & \cdots & a_{3n} \\ \vdots & & \vdots \\ a_{n3} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_3 - a_{31}x_1 - a_{32}x_2 \\ \vdots \\ b_n - a_{n1}x_1 - a_{n2}x_2 \end{bmatrix}. \tag{4.9}$$

In this substitution the values of x_1 and x_2 will not be used but only their symbolic names. Then, x_3 that is a function of x_1 and x_2 can be computed. This process can be continued until all the variables are found and the final expression of each variable will be in the SOE format. This method is also capable of computing for selected variables rather than computing the entire system variables. This is achieved by placing the desired variables at the beginning of the matrix and stopping the computation process when the last variable is obtained. Explicit solutions are also possible by implementing a substitution and evaluation process. Finally, the transfer function between two system variables can efficiently be obtained found by placing these variables at the first two rows and stopping the process after finding the second variable. This algorithm has an exponential time complexity that reduces to quadratic complexity for transfer functions. The key point here is that if x_1 is approximated by device parameter elimination. The error will then propagate to all other variables as can be seen in Equ. 4.8. However, if x_2 is approximated, it will only affect the subsequent variables, x_3, x_4, …, etc. In addition, this error will be scaled by the elements: a_{32}, a_{42}, …, a_{n2}. Thus the effect of each approximation on any system variable can be limited exactly.

4.3. Cancellation Algorithm

Bias circuitry often utilizes a significant portion of the number of devices in an analog integrated circuit. Usually they do not appear in most transfer functions explicitly. Bias device parameters typically factor out of both the numerator and denominator of the transfer functions. The common factors between numerator and denominator are cancelled to obtain a compact and interpretable symbolic transfer function. Less important parameters (such as bias circuitry) are therefore cancelled in this process. The cancellation algorithm operates as follows; for every factor in the expanded numerator, the algorithm checks for a similar factor in the denominator. This is performed by searching for single appearances of terms (singularities) in the factor, and then expanding those singularities in the denominator. If a match is found then a cancellation takes place. This process continues for all numerator factors.

5. Results and Discussion

The proposed symbolic analyzer has been implemented and its capabilities in approximating large formulae are evaluated. The adopted evaluation approach considers the analysis of large networks and performs practical time-complexity analyses. This is achieved by increasing the size of the analyzed circuit and recording the time and memory requirements for each circuit size. A polynomial is then fitted between the network size and the required time (or memory). In this sense, the following implementations of the 741 operational amplifier and the RC ladder networks are used to evaluate performance of the proposed analyzer. The computer system that is used in the analyses of both examples is based on Intel Pentium III processor with 733 MHz clock frequency, 256 MB physical memory, 512 MB virtual memory, and Widows XP operating system.

5.1. Operational Amplifier

Figure 9 shows a typical circuit for the 741 op-amp. This circuit is used here as an example to test the analyzers capability in approximating large formulae, using the proposed shifting approach. It is also used to test the mismatch analysis routines. The assumed mismatching conditions are: (G1, G2), (Q1, Q2), (Q1, Q3), (Q3, Q4), (Q5, Q6), (Q14, Q20), and (G6, G7). The transistor model used in this analysis is the level-one SPICE model with all capacitors being removed. It should be mentioned here that this analysis aims to find a linearized dc gain rather than performing dc analysis of the circuit. The initial number of symbols in the derived dc gain exceeds 400 symbols so that the approximation and shifting techniques are applied with an overall error of less than 10%. The obtained result is found in agreement with that reported in reference [2]. The estimated analysis time is 47 seconds which is consumed mainly in reading and generating the files. The numerical error of the obtained formula is less than 8% (yielding a dc gain of 2.6×10^5) which is improved when compared to a manual computation error of 24% reported in reference [20]. The output formula (during and after the analysis) is kept in SOE format. Such an arrangement minimizes the computation time without significant loss in simplification. The obtained results from this test confirm validity of using the dc voltage gain for formula approximation. This approximation method is frequency independent and therefore the deviation from the nominal point under real operating conditions is minimized.

Further analyses are carried out to evaluate the performance of the mismatch analysis routine. This is achieved by computing the input impedance under the same matched and unmatched conditions. The obtained results in both conditions demonstrate only small deviations from the nominal points. The obtained deviations for the values of $2.0126 \times 10^6 \Omega$ and $2.0376 \times 10^6 \Omega$ are 0.63% (matched

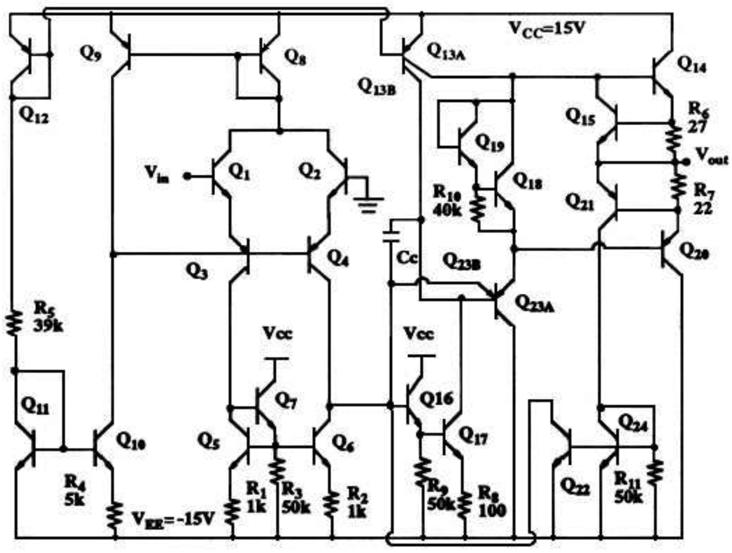


FIGURE 9. Typical circuit of the 741 op-amp

condition) and 1.88% (unmatched condition), respectively. The estimated analysis time in this test is 49 seconds for the matched case and 67 seconds for the unmatched case. When compared with the non-simplified formulae that contain 400 symbolic terms each, the numerical error of the obtained formula is found to be less than 2% in both the matched and unmatched cases.

5.2. RC Ladder Network

This example is used to determine the two-port transmission parameters of an RC passive ladder network that contains 100 sections as shown in Fig. 10. The capability of the proposed analyzer in analyzing large networks can be evaluated in this example. As the difference between using identical and non-identical sections of the ladder circuit is merely notational, the RC sections of this example are chosen to be identical for simplicity. The resulting formulae will also be greatly simplified if only R and C elements are present. Yet, the sizes of matrices will be the same for identical and non-identical sections. Analyses of ladder networks with sections of up to 1000 are also carried out with the aim of determining the time-complexity of the program. A simple polynomial fitting shows that the time complexity is super-quadratic with the number of sections as shown in Fig. 11. A significant reduction in the processing time of this example is demonstrated when implementing the macromodeling techniques. For example, the time required to analyze 1000 ladder sections, using macromodeling, is found to be approximately 1.8 second compared to approximately 451 second in the normal analysis, illustrated in Fig. 11.

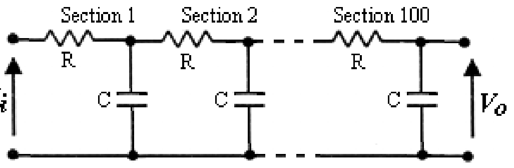


FIGURE 10. Sections of the ladder network

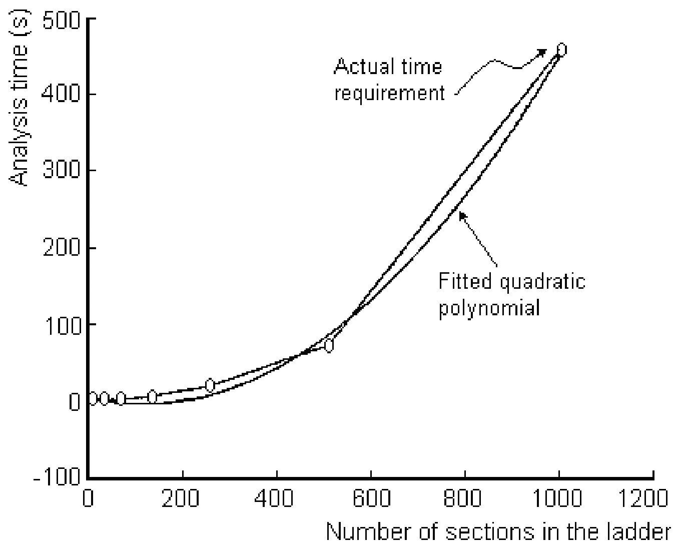


FIGURE 11. Analysis time versus number of sections in the ladder network

6. Conclusions

A symbolic linear analyzer that integrates the analysis of both lumped and distributed networks has been developed and implemented, using a consistent set of notations. The developed analyzer prototype which is based on efficient symbolic matrix solvers, allows the user to derive both transfer functions and formulae. The analyzer performance that is assessed in terms of computation time and storage requirement is further improved by automating the selection of appropriate analysis method. Application examples of a typical 741 op-amp circuit and ladder RC networks are used to assess the overall performance of the developed prototype. The obtained results which are found in agreement with results of previously reported analyses confirm the validity of the proposed formula approximation techniques. When macromodeling techniques are implemented, the analysis time of a network with 1000 ladder sections is found to be only a small fraction (less than 0.4%) of that required in the normal analysis. Finally, compatibility of the proposed analyzer with the MATLAB makes it possible to use all features and benefits offered by the MATLAB package as well.

References

[1] G.E. Gielen, H. Walscharts and W. Sansen, *ISAAC — A symbolic simulator for analog integrated circuits.* IEEE Trans. Solid-State Circuits **24** (1989), 1587–1597.

[2] J.J. Hsu and C. Sechen, *DC small signal symbolic analysis oflarge analog integrated circuits.* IEEE Trans. on Circuits and Systems-Fundamental Theory and Applications **41** (1994), 817–828.

[3] O. Guerra, E. Roca, E.V. Fernandez and A. Rodriguez-Vazquez, *Arhierarchical approach for the symbolic analysis of large analogue integrated circuits.* IEEE Conf. Proc. on Design, Automation, and Test in Europe (DATE00), France (2000), 48–52.

[4] C.-J. R. Shi and X.-D. Tan, *Canonical symbolic analysis of large analog circuits with determinant decision diagrams.* IEEE Trans. Computer-Aided Design **19** (2000), 1–18.

[5] X.-D. Tan and C.-J. Shi, *Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams.* IEEE Trans. Computer-Aided Design **19** (2000), 401–412.

[6] F. Fernandez and A. Rodriguez-Vazquez, *Symbolic analysis tools-the state of the art.* Proc. IEEE Int. Symp. on Circuits and Systems (1996), 798–801.

[7] F. Fernandez, A. Rodriguez-Vazquez, J. Huertas and G. Gielen, *Symbolic Analysis Techniques.* IEEE Press (1998).

[8] Q. Yu and C. Sechen, *A unified approach to the approximate symbolic analysis of large analog integrated circuits.* IEEE Trans. Circuits and Systems **43** (1996), 656–669.

[9] P. Wambacq, G. Gielen and W. Sansen, *A new reliable approximation method for expanded symbolic network functions.* Proc. IEEE Int. Symp. on Circuits and Systems (1996), 584–587.

[10] C.-J. R. Shi and X.-D. Tan, *Compact representation and efficient generation of s-expanded symbolic network functions for computer-aided analog circuit design.* IEEE Trans. Computer-Aided Design **20** (2001), 813–827.

[11] M.M. Hassoun and P.-M. Lin, *A hierarchical network approach to symbolic analysis of large scale networks.* IEEE Trans. on Circuits and Systems **42** (1995), 201–211.

[12] M. Monagan, *Programming in Maple: The Basics.* Institute für Wissenschaftliches Rechnen, Zürich (1997).

[13] J.M. Ortega and A.S. Grimshaw, *An Introduction to C++ and Numerical Methods.* Oxford University Press (1999).

[14] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* Van Nostrand Reinhold (1994).

[15] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press (1992).

[16] P. Sannuti and N.N. Puri, *Symbolic network analysis: an algebraic formulation.* IEEE Trans. on Circuits and Systems **27** (1980), 679–687.

[17] W.J. Gilbert and W.K. Nicholson, *Modern Algebra with Applications.* John Wiley and Sons (2004).

[18] V. Shoup, *A computational introduction to number theory and algebra (V.1).* www.shoup.net/ntb (2005).

[19] C.-J. Shi and X.-D. Tan, *Symbolic analysis of large analog circuits with determinant decision diagrams.* IEEE/ACM Proc. Int. Conf. on Computer-Aided Design (1997), 366–373.

[20] P.R. Gray and R.G. Meyer, *Analysis and Design of Analog Integrated Circuits.* John Wiley and Sons (1993).

Majid A. Al-Taee
Computer Engineering Department
The University of Jordan
Amman 11942 Jordan
e-mail: `altaeem@ju.edu.jo`

Prof. Fawzi M. Al-Naima
Department of Computer Engineering
College of Engineering, Al-Nahrain University
PO Box 64040 Jadriya
Baghdad, IRAQ
e-mail: `fawzialnaima@yahoo.com`
e-mail: `falnaima@cmc.iq`
current address on Sabbatical Leave:
College of Electrical and Electronic Engineering
Aleppo University
University Street
Aleppo, Syria
`http://www.alepuniv@shern.net`

Dr. Bessam Al-Jewad
Department of Computer Engineering
College of Engineering
Al-Nahrain University
PO Box 64040 Jadriya
Baghdad, IRAQ
e-mail: `bessam.aljewad@atsiraq.com`

Author Index