Lecture Notes in Physics

The Editorial Policy for Monographs

The series Lecture Notes in Physics reports new developments in physical research and teaching - quickly, informally, and at a high level. The type of material considered for publication includes monographs presenting original research or new angles in a classical field. The timeliness of a manuscript is more important than its form, which may be preliminary or tentative. Manuscripts should be reasonably self-contained. They will often present not only results of the author(s) but also related work by other people and will provide sufficient motivation, examples, and applications.

Acceptance

The manuscripts or a detailed description thereof should be submitted either to one of the series editors or to the managing editor. The proposal is then carefully refereed. A final decision concerning publication can often only be made on the basis of the complete manuscript, but otherwise the editors will try to make a preliminary decision as definite as they can on the basis of the available information.

Contractual Aspects

Authors receive jointly 30 complimentary copies of their book. No royalty is paid on Lecture Notes in Physics volumes. But authors are entitled to purchase directly from Springer other books from Springer (excluding Hager and Landolt-Börnstein) at a $33\frac{1}{3}$% discount off the list price. Resale of such copies or of free copies is not permitted. Commitment to publish is made by a letter of interest rather than by signing a formal contract. Springer secures the copyright for each volume.

Manuscript Submission

Manuscripts should be no less than 100 and preferably no more than 400 pages in length. Final manuscripts should be in English. They should include a table of contents and an informative introduction accessible also to readers not particularly familiar with the topic treated. Authors are free to use the material in other publications. However, if extensive use is made elsewhere, the publisher should be informed. As a special service, we offer free of charge LaTeX macro packages to format the text according to Springer's quality requirements. We strongly recommend authors to make use of this offer, as the result will be a book of considerably improved technical quality. The books are hardbound, and quality paper appropriate to the needs of the author(s) is used. Publication time is about ten weeks. More than twenty years of experience guarantee authors the best possible service.

LNP Homepage (springerlink.com)

On the LNP homepage you will find:
−The LNP online archive. It contains the full texts (PDF) of all volumes published since 2000. Abstracts, table of contents and prefaces are accessible free of charge to everyone. Information about the availability of printed volumes can be obtained.
−The subscription information. The online archive is free of charge to all subscribers of the printed volumes.
−The editorial contacts, with respect to both scientific and technical matters.
−The author's / editor's instructions.

Dieter Britz

Digital Simulation in Electrochemistry

Third Completely Revised and Extended Edition
With Supplementary Electronic Material

Springer

Author

Dieter Britz
Kemisk Institut
Århus Universitet
8000 Århus C
Denmark
Email: britz@chem.au.dk

Dieter Britz, *Digital Simulation in Electrochemistry*,
Lect. Notes Phys. **666** (Springer, Berlin Heidelberg 2005), DOI 10.1007/b97996

Library of Congress Control Number: 2005920592

ISSN 0075-8450
ISBN 3-540-23979-0 3rd ed. Springer Berlin Heidelberg New York

ISBN 3-540-18979-3 2nd ed. Springer-Verlag Berlin Heidelberg New York
ISBN 3-540-10564-6 1st ed. published as Vol. 23 in *Lecture Notes in Chemistry*
Springer-Verlag Berlin Heidelberg New York

Typesetting: by the authors and TechBooks using a Springer LaTeX macro package
Cover design: *design & production*, Heidelberg

Printed on acid-free paper
2/3141/jl - 5 4 3 2 1 0

This book is dedicated to H. H. Bauer, teacher and friend

Preface

This book is an extensive revision of the earlier 2nd Edition with the same title, of 1988. The book has been rewritten in, I hope, a much more didactic manner. Subjects such as discretisations or methods for solving ordinary differential equations are prepared carefully in early chapters, and assumed in later chapters, so that there is clearer focus on the methods for partial differential equations. There are many new examples, and all programs are in Fortran 90/95, which allows a much clearer programming style than earlier Fortran versions.

In the years since the 2nd Edition, much has happened in electrochemical digital simulation. Problems that ten years ago seemed insurmountable have been solved, such as the thin reaction layer formed by very fast homogeneous reactions, or sets of coupled reactions. Two-dimensional simulations are now commonplace, and with the help of unequal intervals, conformal maps and sparse matrix methods, these too can be solved within a reasonable time. Techniques have been developed that make simulation much more efficient, so that accurate results can be achieved in a short computing time. Stable higher-order methods have been adapted to the electrochemical context.

The book is accompanied (on the webpage www.springerlink.com/openurl.asp?genre=issue&issn=1616-6361&volume=666) by a number of example procedures and programs, all in Fortran 90/95. These have all been verified as far as possible. While some errors might remain, they are hopefully very few.

I have a debt of gratitude to a number of people who have checked the manuscript or discussed problems with me. My wife Sandra polished my English style and helped with some of the mathematics, and Tom Koch Svennesen checked many of the mathematical equations. Others I have consulted for advice of various kinds are Professor Dr. Bertel Kastening, Drs. Lesław Bieniasz, Ole Østerby, Jörg Strutwolf and Thomas Britz. I thank the various editors at Springer for their support and patience. If I have left anybody out, I apologize. As is customary to say (and true), any errors remaining in the book cannot be blamed on anybody but myself.

Århus, *Dieter Britz*
February 2005

Contents

1 Introduction

This book is about the application of digital simulation to electrochemical problems. What is digital simulation? The term "simulation" came into wide use with the advent of analog computers, which could produce electrical signals that followed mathematical functions to describe or model a given physical system. When digital computers became common, people began to do these simulations digitally and called this digital simulation. What sort of systems do we simulate in electrochemistry? Most commonly they are electrochemical transport problems that we find difficult to solve, in all but a few model systems – when things get more complicated, as they do in real electrochemical cells, problems may not be solvable algebraically, yet we still want answers.

Most commonly, the basic equation we need to solve is the diffusion equation, relating concentration c to time t and distance x from the electrode surface, given the diffusion coefficient D:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} \; . \tag{1.1}$$

This is Fick's second diffusion equation [242], an adaptation to diffusion of the heat transfer equation of Fourier [253]. Technically, it is a second-order parabolic partial differential equation (*pde*). In fact, it will mostly be only the skeleton of the actual equation one needs to solve; there will usually be such complications as convection (solution moving) and chemical reactions taking place in the solution, which will cause concentration changes in addition to diffusion itself. Numerical solution may then be the only way we can get numbers from such equations – hence digital simulation.

The numerical technique most commonly employed in digital simulation is (broadly speaking) that of finite differences and this is much older than the digital computer. It dates back at least to 1911 [468] (Richardson). In 1928, Courant, Friedrichs and Lewy [182] described what we now take to be the essentials of the method; Emmons [218] wrote a detailed description of finite difference methods in 1944, applied to several different equation types. There is no shortage of mathematical texts on the subject: see, for example, Lapidus and Pinder [350] and Smith [514], two excellent books out of a large number.

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 1–4 (2005)
www.springerlink.com

It should not be imagined that the technique became used only when digital computers appeared; engineers certainly used it long before that time, and were not afraid to spend hours with pencil and paper. Emmons [218] casually mentions that one fluid flow problem took him 36 hours! Not surprisingly, it was during this early pre-computer era that much of the theoretical groundwork was laid and refinements worked out to make the work easier – those early stalwarts wanted their answers as quickly as possible, and they wanted them correct the first time through.

Electrochemical digital simulation is almost synonymous with Stephen Feldberg, who wrote his first paper on it in 1964 [234]. It is not always remembered that Randles [460] used the technique much earlier (in 1948), to solve the linear sweep problem. He did not have a computer and did the arithmetic by hand. The most widely quoted electrochemical literature source is Feldberg's chapter in Electroanalytical Chemistry [229], which describes what will here be called the "box" method. Feldberg is rightly regarded as the pioneer of digital simulation in electrochemistry, and is still prominent in developments in the field today. This has also meant that the box method has become standard practice among electrochemists, while what will here be called the "point" method is more or less standard elsewhere. Having experimented with both, the present author favours the point method for the ease with which one arrives at the discrete form of one's equations, especially when the differential equation is complicated.

A brief description will now be given of the essentials of the simulation technique. Assume (1.1) above. We wish to obtain concentration values at a given time over a range of distances from the electrode. We divide space (the x coordinate) into small intervals of length h and time t into small time steps δt. Both x and t can then be expressed as multiples of h and δt, using i as the index along x and j as that for t, so that

$$x_i = ih \tag{1.2}$$

and

$$t_j = j\delta t\,. \tag{1.3}$$

Figure 1.1 shows the resulting grid of points. At each drawn point, there is a value of c. The digital simulation method now consists of developing rows of c values along x, (usually) one t-step at a time. Let us focus on the three filled-circle points c_{i-1}, c_i and c_{i+1} at time t_j. One of the various techniques to be described will compute from these three known points a new concentration value $c'_i = c_i(t = (j+1)\delta t)$ (empty circle) at x_i for the next time value t_{j+1}, by expressing (1.1) in discrete form:

$$\frac{c'_i - c_i}{\delta t} = \frac{D}{h^2}\left(c_{i-1} - 2c_i + c_{i+1}\right). \tag{1.4}$$

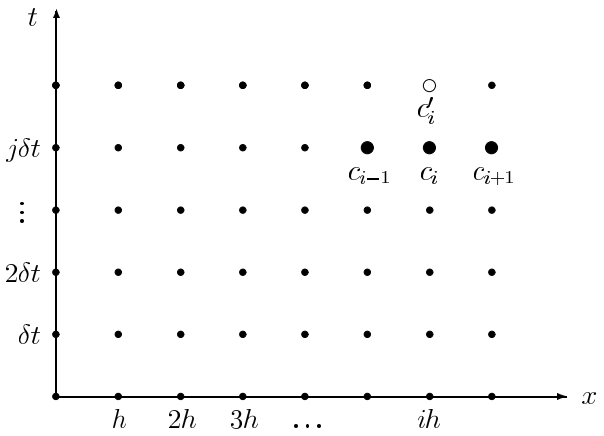
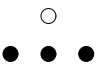


Fig. 1.1. Discrete sample point grid

The only unknown in this equation is c'_i and it can be explicitly calculated. Having obtained c'_i, we move on to the next x point and compute c' for it, etc., until all c values for that row, for the next time value, have been computed.

In the remainder of the book, the various schemes for calculating new points will often be graphically described by isolating the marked circles seen in Fig. 1.1; in this case, the scheme would be represented by the following diagram



This follows the convention seen in such texts as Lapidus and Pinder [350] (who call it the "computational molecule", which will also be the name for it in this book). It is very convenient, as one can see at a glance what a particular scheme does. The filled points are known points while the empty circles are those to be calculated.

Several problems will become apparent. The first one is that of the method used to arrive at (1.4); this will be dealt with later. There is, in fact, a multiplicity of methods and expressions used. The second problem is the concentration value at $x = 0$; there is no x_{-1} point, as would be needed for $i = 1$. The value of c_0 is a *boundary value*, and must be determined by some other method. Another boundary value is the last x point we treat. How far out into the diffusion space should (need) we go? Usually, we know good approximations for concentrations at some sufficiently large distance from the electrode (e.g., either "bulk" concentration, or zero for a species generated at the electrode), and we have pretty good criteria for the distance we need to go out to. Another boundary lies at the row for $t = 0$: this is the row of starting values. Again, these are supplied by information other than

the diffusional process we are simulating (but, for a given method, can be a problem, as will be seen in a later chapter). Boundary problems are dealt with in Chap. 6. They are, in fact, a large part of what this book is about, or what makes it specific to electrochemistry. The discrete diffusion equation we have just gone through could just as well apply to heat transfer or any other diffusion transport problems.

Throughout the book, the following symbol convention will be used: dimensioned quantities like concentration, distance or time will be given lowercase symbols (c, x, t, etc.) and their non-dimensional equivalents will be given the corresponding upper-case symbols (C, X, T, etc.), with a few unavoidable exceptions.

2 Basic Equations

2.1 General

In this chapter, we present most of the equations that apply to the systems and processes to be dealt with later. Most of these are expressed as equations of concentration dynamics, that is, concentration of one or more solution species as a function of time, as well as other variables, in the form of differential equations. Fundamentally, these are transport (diffusion-, convection- and migration-) equations but may be complicated by chemical processes occurring heterogeneously (i.e. at the electrode surface – electrochemical reaction) or homogeneously (in the solution bulk – chemical reaction). The transport components are all included in the general Nernst-Planck equation (see also Bard and Faulkner 2001) for the flux J_j of species j

$$\boldsymbol{J}_j = -D_j \boldsymbol{\nabla} C_j - \frac{z_j \mathcal{F}}{\mathcal{R}\mathcal{T}} D_j C_j \boldsymbol{\nabla}\phi + C_j \boldsymbol{v} \tag{2.1}$$

in which $\boldsymbol{J}_j$ is the molar flux per unit area of species j at the given point in space, D_j the species' diffusion coefficient, C_j its concentration, z_j its charge, $\mathcal{F}$, $\mathcal{R}$ and $\mathcal{T}$ have their usual meanings, ϕ is the potential and $\boldsymbol{v}$ the fluid velocity vector of the surrounding solution (medium). The symbol $\boldsymbol{\nabla}$ denotes the differentiation operator and it is directional in 3-D space. This equation is a more general form of Fick's first diffusion equation, which contains only the first term on the right-hand side, the diffusion term. The second term on that side is the migration term and the last, the convection term. These will now be discussed individually. At the end of the chapter, we go through some models and electrode geometries, and present some known analytical solutions, as well as dimensionless forms of the equations. There is no term in the equation to take account of changes due to chemical reactions taking place in the solution, since these do not give rise to a flux of substance. Such terms come in later, in the equations relating concentration changes with time to the above components (see (2.15) and Sect. 2.2.6).

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 5–32 (2005)
www.springerlink.com

2.2 Some Mathematics: Transport Equations

2.2.1 Diffusion

For a good text on diffusion, see the monograph of Crank [183]. Consider Fig. 2.1. We imagine a chosen coordinate direction x in a solution volume containing a dissolved substance at concentration c, which may be different at different points – i.e., there may be concentration gradients in the solution. We consider a very small area δA on a plane normal to the x-axis. Fick's first equation now says that the net flow of solute (flux f_x, in $\mathrm{mol\,s^{-1}}$) crossing the area is proportional to the negative of the concentration gradient at the plane, in the x-direction

$$f_x = \frac{\mathrm{d}n}{\mathrm{d}t} = -\delta A\, D \frac{\mathrm{d}c}{\mathrm{d}x} \tag{2.2}$$

with D a proportionality constant called the diffusion coefficient and n the number of moles. This can easily be understood upon a moment's thought; statistically, diffusion is a steady spreading out of randomly moving particles. If there is no concentration gradient, there will be an equal number per unit time moving backward and forward across the area δA, and thus no net flow. If there is a gradient, there will be correspondingly more particles going in one direction (down the gradient) and a net increase in concentration on the lower side will result. Equation (2.2) is of precisely the same form as the first heat flow equation of Fourier [253]; Fick's contribution [242] lay in realising the analogy between temperature and concentration, heat and mass (or number of particles). The quantity D has units $\mathrm{m^2 s^{-1}}$ (SI) or $\mathrm{cm^2 s^{-1}}$ (cgs).

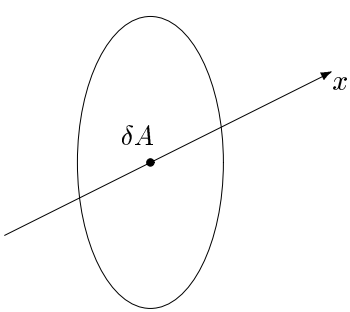


Fig. 2.1. Diffusion across a small area

Equation (2.2) is the only equation needed when using the box method and this is sometimes cited as an advantage. It brings one close to the microscopic system, as we shall see, and has – in theory – great flexibility in cases where the diffusion volume has an awkward geometry. In practice, however, most geometries encountered will be – or can be simplified to – one of but a few

standard forms such as cartesian, cylindrical or spherical – for which the full diffusion equation has been established (see, e.g., Crank [183]). In cartesian coordinates this equation, Fick's second diffusion equation, in its most general form, is

$$\frac{\partial c}{\partial t} = D_x \frac{\partial^2 c}{\partial x^2} + D_y \frac{\partial^2 c}{\partial y^2} s + D_z \frac{\partial^2 c}{\partial z^2} \ . \tag{2.3}$$

This expresses the rate of change of concentration with time at given coordinates (t, x, y, z) in terms of second space derivatives and three different diffusion coefficients. It is theoretically possible for D to be direction-dependent (in anisotropic media) but for a solute in solution, it is equal in all directions and usually the same everywhere, so (2.3) simplifies to

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2} \right) \ , \tag{2.4}$$

that is, the usual three-dimensional form. Even this is rather rarely applied – we always try to reduce the number of dimensions, preferably to one, giving

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} \ . \tag{2.5}$$

If the geometry of the system is cylindrical, it is convenient to switch to cylindrical coordinates: x along the cylinder, r the radial distance from the axis and θ the angle. In most cases, concentration is independent of the angle and the diffusion equation is then

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial r^2} + \frac{1}{r} \frac{\partial c}{\partial r} \right) \ . \tag{2.6}$$

Often there is no gradient along x (the axis), so only r remains

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial r^2} + \frac{1}{r} \frac{\partial c}{\partial r} \right) \ . \tag{2.7}$$

For a spherical system, assuming no concentration gradients other than away from the centre (radially), the equation becomes

$$\frac{\partial c}{\partial t} = D \left(\frac{\partial^2 c}{\partial r^2} + \frac{2}{r} \frac{\partial c}{\partial r} \right) \ . \tag{2.8}$$

2.2.2 Diffusion Current

Equation (2.2) gives the flux in mol s^{-1} of material as the result of a concentration gradient. If there is such a gradient normal to an electrode/electrolyte interface, then there is a flux of material at the electrode and this takes place via the electron transfer. An electroactive species diffuses to the electrode,

takes part in the electron transfer and becomes a new species. The electrical current i flowing is then equal to the molar flux multiplied by the number of electrons transferred for each molecule or ion (2.2), and the Faraday constant

$$i = nFAD \left(\frac{\partial c}{\partial x} \right)_{x=0} \tag{2.9}$$

for a reduction current. The flux and the current are thus, in a sense, synonymous and will, in fact, profitably be expressed simply in terms of the concentration gradient itself or its dimensionless equivalent, to be discussed later (Sect. 2.3).

2.2.3 Convection

If we cannot arrange for our solution to be (practically) stagnant during our experiment, then we must include convective terms in the equations. Figure 2.2 shows a plot of concentration against the x-coordinate at a given instant. Let x_1 be a fixed point along x, with concentration c_1 at some time t, and let the solution be moving forward along x with velocity v_x, so that after a small time interval δt, concentration c_2 (previously at x_2) has moved to x_1 by the distance δx. If δt and δx are chosen sufficiently small, we may consider the line PQ as straight and we have, for the change δc at x_1

$$\delta c = -\delta x \frac{\mathrm{d}c}{\mathrm{d}x} \tag{2.10}$$

Dividing by δt, taking $v_x = \delta x / \delta t$ and going to the infinitesimal limit, we get for the x-term

$$\frac{\partial c}{\partial t} = -v_x \frac{\partial c}{\partial x} \,. \tag{2.11}$$

If there is convection in all three directions, this expands to

$$\frac{\partial c}{\partial t} = -v_x \frac{\partial c}{\partial x} - v_y \frac{\partial c}{\partial y} - v_z \frac{\partial c}{\partial z} \,. \tag{2.12}$$

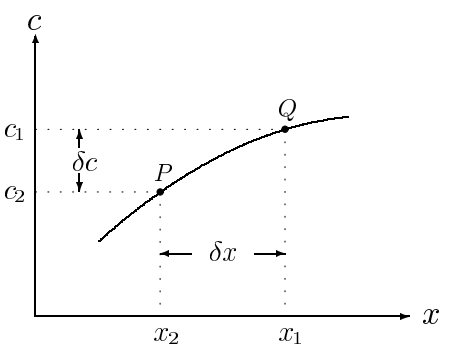


Fig. 2.2. Convection

This treatment ignores the diffusional processes taking place simultaneously; the two transport terms are additive in the limit.

Convection terms commonly crop up with the dropping mercury electrode, rotating disk electrodes and in what has become known as hydrodynamic voltammetry, where the electrolyte is made to flow past an electrode in some reproducible way (e.g. the impinging jet, channel and tubular flows, vibrating electrodes, etc). This is discussed in Chap. 13.

2.2.4 Migration

Migration is included here more or less for completeness – the electrochemist is usually able to eliminate this transport term (and will do so for practical reasons as well). If our species is charged, that is, it is an ion, then it may experience electrical forces due to potential fields. This will be significant in solutions of ionic electroactive species, not containing a sufficiently large excess of inert electrolyte.

In general (see Vetter [559]), for an electroactive cation with charge $+z_A$ and anion with charge $-z_B$, an inert electrolyte with the same two charges on its ions, and with r the concentration ratio electrolyte/electroactive ion, we have the rather awkward equation

$$\frac{i}{i_0} = \left(1 + \left|\frac{z_A}{z_B}\right|\right)(1+r)\left(1 - \left(\frac{r}{1+r}\right)^p\right) \tag{2.13}$$

where

$$p = \left(1 + \left|\frac{z_A}{z_B}\right|\right)^{-1} \tag{2.14}$$

and i_0 is the pure diffusion current, without migration effects. To illustrate, let us take $|z_A| = |z_B| = 1$. Then $i/i_0 = 2$ for $r = 0$ (no inert electrolyte), 1.17 for $r = 1$, 1.02 for $r = 10$ and 1.002 for $r = 100$. For very accurate studies, then, inert electrolyte should be in excess by a factor of 100 or more, and this will be assumed in the remainder of the book.

There is one situation in which migration can have an appreciable effect, even in the presence of excess inert electrolyte. For the measurement of very fast reactions, one must resort to techniques involving very small diffusion layers (see Sect. 2.4.1 for the definition) – either by taking measurements at very short times or forcing the layer thickness down by some means. If that thickness becomes comparable in magnitude with that of the diffuse double layer, and the electroactive species is charged, then migration will play a part in the transport to and from the electrode. The effect has been clearly explained elsewhere [83]. A rough calculation for a planar electrode in a stagnant solution, assuming the thickness of the diffuse double layer to be of the order of 10^{-9}m and the diffusion coefficient of the electroactive species to be $10^{-12}\,\mathrm{m^2 s^{-1}}$ (which is rather slow) shows that migration effects are expected during the first μs or so. The situation, then, is rather extreme and

we leave it to the specialist to handle it. Recently, this has been discussed [513] in the context of ultramicroelectrodes, where this may need to be investigated further.

2.2.5 Total Transport Equation

This section serves merely to emphasise that for a given cell system, the full transport equation is the sum of those for diffusion, convection and migration. We might write, quite generally,

$$\frac{\partial c}{\partial t} = \left(\frac{\partial c}{\partial t}\right)_{\text{diff}} + \left(\frac{\partial c}{\partial t}\right)_{\text{conv}} + \left(\frac{\partial c}{\partial t}\right)_{\text{migr}} \tag{2.15}$$

with the "diff" term as defined by one of the (2.3)–(2.8), the "conv" term by (2.11) and "migr" related to (2.13). At any one instant, these terms are simply additive. Digitally, we can "freeze" the instant and evaluate the sum of the separate terms. There may be non-transport terms to add as well, such as kinetic terms, to be discussed next.

2.2.6 Homogeneous Kinetics

Homogeneous reactions are chemical reactions not directly dependent upon the electrode/electrolyte interface, taking place somewhere within the electrolyte (or, in principle, the metal) phase. These lead to changes in concentration of reactants and/or products and can have marked effects on the dynamics of electrochemical processes. They also render the dynamic equations much more difficult to solve and it is here that digital simulation sees much of its use. Whereas analytical solutions for kinetic complications are difficult to obtain, the corresponding discrete expressions are obtained simply by extending the diffusion equation by an extra, kinetic term (although practical problems arise, see Chaps. 5, 9). The actual form of this depends upon the sort of chemistry taking place. In the simplest case, met with in flash photolysis, we have a single substance generated by the flash, then decaying in solution by a first- or second-order reaction; this is represented by equations of the form

$$\frac{\partial c}{\partial t} = -k_1 c \tag{2.16}$$

or

$$\frac{\partial c}{\partial t} = -2k_2 c^2 \tag{2.17}$$

and these can be added to the transport terms. Very often, we have several substances interacting chemically, as in the example of the simple electrochemical reaction

$$A + ne^- \Leftrightarrow B \tag{2.18}$$

followed by chemical decay of the product B. If this is first-order and we have a simple one-dimensional diffusion system, we then have the two equations (c_A and c_B denoting concentrations of, substances A and B, respectively; D_A and D_B the two respective diffusion coefficients)

$$\begin{aligned}\frac{\partial c_A}{\partial t} &= D_A \frac{\partial^2 c_A}{\partial x^2} \\ \frac{\partial c_B}{\partial t} &= D_B \frac{\partial^2 c_B}{\partial x^2} - k_1 c_B \ .\end{aligned} \tag{2.19}$$

There is a great variety of such reactions including dimerisation, disproportionation and catalytic reactions, both preceding and following the electrochemical step(s) and it is not useful to attempt to list them all here. The point is merely to stress that they are (with greater or lesser difficulty) digitally tractable, as will be shown in Chaps. 5 and 9.

There is one problem that makes homogeneous chemical reactions especially troublesome. Most often, a mechanism to be simulated involves species generated at the interface, that then undergo chemical reaction in the solution. This leads to concentration profiles for these species that are confined to a thin layer near the interface – thin, that is, compared with the diffusion layer (see Sect. 2.4.1, the Nernst diffusion layer). This is called the reaction layer (see [74, 257, 559]). Simulation parameters are usually chosen so as to resolve the space within the diffusion layer and, if a given profile is much thinner than that, the resolution of the sample point spacing might not be sufficient. The thickness of the reaction layer depends on the nature of the homogeneous chemical reaction. In any case, any number given for such a thickness – as with the diffusion layer thickness – depends on how the thickness is defined. Wiesner [572] first derived an expression for the reaction layer thickness μ,

$$\mu = \sqrt{\frac{D}{k}} \ . \tag{2.20}$$

(Wiesner's expression used different symbols, but this is not important.) This expression strictly holds only for a first-order reaction and Vetter [559] provides a more general expression. However, the above expression is sufficient for most simulation purposes. The equation for μ holds in practice only for rather large values of the rate constant; for small values below unity, μ becomes greater than the diffusion layer thickness, which will then dominate the concentration profile. At the other end of the scale of rate constants, for very fast reactions, μ can become very small. The largest rate constant possible is about $10^{10}\mathrm{s}^{-1}$ (the diffusion limit) and this leads to a μ value only about 10^{-5} the thickness of the diffusion layer, so there must be some sample points very close to the electrode. This problem has been overcome only in recent years, first by using unequal intervals, then by the use of dynamic grids, both of which are discussed in Chap. 7.

2.2.7 Heterogeneous Kinetics

In real (as opposed to model) electrochemical cells, the net current flowing will often be partly determined by the kinetics of electron transfer between electrode and the electroactive species in solution. This is called heterogeneous kinetics, as it refers to the interface instead of the bulk solution. The current in such cases is obtained from the Butler-Volmer expressions relating current to electrode potential [73,74,83,257,559]. We have at an electrode the process (2.18), with concentrations at the electrode/electrolyte interface $c_{A,0}$ and $c_{B,0}$, respectively. We take as positive current that going into the electrode, i.e., electrons leaving it, which corresponds to the reaction (2.18) going from left to right, or a reduction. Positive or forward (reduction) current i_f is then related to the potential E by

$$i_f = nFAc_{A,0}k_0 \exp\left(\frac{-\alpha n\mathcal{F}}{\mathcal{RT}}\left(E - E^0\right)\right) \tag{2.21}$$

with A the electrode area, k_0 a standard heterogeneous rate constant, α the so-called transfer coefficient which lies between 0 and 1 and E^0 the system's standard potential. For the reverse (oxidation) current i_b,

$$i_b = -nFAc_{B,0}k_0 \exp\left((1-\alpha)\frac{n\mathcal{F}}{\mathcal{RT}}\left(E - E^0\right)\right) . \tag{2.22}$$

Both processes may be running simultaneously. The net current is then the sum $(i_f + i_b)$ and this will, through (2.9), fix the concentration gradients at the electrode in these cases.

If a reaction is very fast, it may be simpler to make the assumption of complete reversibility or electrochemical equilibrium at the electrode, at a given potential E. The Nernst equation then applies:

$$E = E^0 - \frac{\mathcal{RT}}{n\mathcal{F}} \ln\left(\frac{c_{B,0}}{c_{A,0}}\right) \tag{2.23}$$

or, for the purpose of computation,

$$\frac{c_{A,0}}{c_{B,0}} = \exp\left(\frac{n\mathcal{F}}{\mathcal{RT}}\left(E - E^0\right)\right) . \tag{2.24}$$

Just how this is applied in simulation will be seen in later chapters.

The foregoing ignores activity coefficients. If these are known, they can be inserted. Most often they are taken as unity.

2.3 Normalisation – Making the Variables Dimensionless

In most simulations, it will be advantageous to transform the given equation variables into dimensionless ones. This is done by expressing them each as a

multiple of a chosen reference value, so that they no longer have dimensions. The time variable t, for example, is expressed as a multiple of some characteristic time τ, which may be different things depending upon the experiment to be simulated. Sometimes it might be the total duration of an experiment (the observation time) or, in the case of a linear sweep experiment, the length of time it takes for the voltage to change by some specified amount. The distance from an electrode x can be conveniently expressed as a multiple of some characteristic distance δ, which will be defined below. Concentrations are normally expressed as multiples of some reference concentration, usually the initial bulk concentration of a certain species involved in the reaction, say c^*. The convention adopted in the rest of the book is, then, that the new dimensionless variables, written in capitals, are

$$\begin{aligned} C &= c/c^* \\ X &= x/\delta \\ T &= t/\tau \,. \end{aligned} \tag{2.25}$$

The reference time scale τ depends on the system to be simulated, as will be seen in the next section, where some model systems are described. There, the characteristic distance δ will also be defined as used in this book (Sect. 2.4.1). Other variables that are normalised are the current and electrode potential. Current i is proportional to the concentration gradient, by Fick's first equation (2.2), as expressed in (2.9). We introduce the dimensionless gradient or flux, defined as

$$G = \left.\frac{\partial C}{\partial X}\right|_{X=0} \,. \tag{2.26}$$

This will now represent the current in dimensionless form. The actual current can be calculated by denormalisation, that is,

$$i = n\,F\,A\,D\,G\,\frac{c^*}{\delta} \,. \tag{2.27}$$

The standard heterogeneous rate constant, seen in (2.21) and (2.22), with dimensions $\mathrm{m\,s^{-1}}$, is normalised by

$$K_0 = k_0\sqrt{\tau/D} \,. \tag{2.28}$$

Potential values (in V) are normalised by the $\mathcal{RT}/n\mathcal{F}$ unit and usually by referring to some reference value E^0, using (in this book) the symbol p:

$$p = \frac{n\mathcal{F}}{\mathcal{RT}}\left(E - E^0\right) \tag{2.29}$$

so that one p-unit corresponds to $25.69\,n\,\mathrm{mV}$. Thus, the two Butler-Volmer components in (2.21) and (2.22) can be expressed in terms of dimensionless current G as

$$G = K_f C_{A,0} - K_b C_{B,0} \tag{2.30}$$

with

$$\begin{aligned} K_f &= K_0 \exp\{-\alpha p\} \\ K_b &= K_0 \exp\{(1-\alpha) p\} \end{aligned} \tag{2.31}$$

and the Nernst equation very simply as

$$\frac{C_{A,0}}{C_{B,0}} = e^p \,. \tag{2.32}$$

With certain rules and tricks, as will be shown, this will lead to equations whose solutions are much more general and useful than if we solve the dimensioned equation for our particular parameter set of values.

2.4 Some Model Systems and Their Normalisations

When developing a new simulation method, it is good to have a number of model systems at hand, for which there are known results, whether these be in the form of analytical solutions (concentration profiles, current) or well established series solutions (as in the case of linear sweep voltammetry, where some parameters have been calculated to quite high precision). The test models should be chosen, as far as possible, to challenge the method. If the new method's primary purpose, for example, is simply greater efficiency, then a simple model like the Cottrell system and chronopotentiometry may be enough to demonstrate that; these two differ fundamentally in their boundary conditions, the Cottrell system having a so-called Dirichlet boundary condition (given concentrations at the boundary), while chronopotentiometry has a derivative or Neumann condition, where gradients are specified at the boundary. If a method under development is expected to give high resolution (small intervals) along x – usually at the boundary – a model that provides marked concentration changes very close to the boundary is the best for testing that.

Along with a group of models that have shown themselves useful, their particular normalisations will be presented. The first model, the Cottrell system, will also serve to introduce the concept of the Nernst diffusion layer.

2.4.1 Potential Steps

Potential step experiments are a popular way to look at electrochemical kinetics. The oldest known is the Cottrell system, where the potential stepped to is so far negative that the resulting current is limited by the transport of the active substance. If the step is not so far negative, one then has either

Nernstian boundary conditions, or those for quasireversible or irreversible systems. All of these cases have been analytically solved. As well, there are two systems involving homogeneous chemical reactions, from flash photolysis experiments, for which there exist solutions to the potential step experiment, and these are also given; they are valuable tests of any simulation method, especially the second-order kinetics case.

Cottrell System

We introduce here the diffusion-controlled potential-step experiment, hereafter called the Cottrell experiment [181]. Consider Fig. 2.3, showing a long thin tube representing an electrochemical cell, bounded at one end by an electrode and filled with electrolyte and an electroactive substance initially at concentration c^* (the bulk concentration). We place the electrode at $x = 0$ and the other, counter-electrode (not shown), at a large distance so that what happens there is of no consequence to us. We apply, at $t = 0$, a potential such that our electroactive substance reacts at the electrode infinitely fast – that is, its concentration c_0 at the electrode ($x = 0$) is forced to zero and kept there.

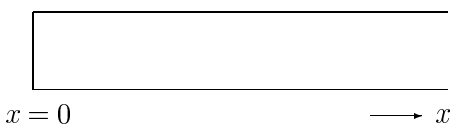


Fig. 2.3. A semi-infinite one-dimensional cell

Clearly, there will be flow of substance towards the electrode by diffusion (we assume no convection here) and we will gradually cause some depletion of material in the solution near $x = 0$; this depletion region will grow out from the electrode with time. Mathematically, this is described by the diffusion equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} \tag{2.33}$$

with the boundary conditions

$$\begin{aligned} t < 0, \text{all } x: &\quad c = c^* \\ t \geq 0, x = 0: &\quad c = 0 \\ \text{all } t, x \to \infty: &\quad c = c^* \ . \end{aligned} \tag{2.34}$$

This classical equation with the boundary conditions as shown has an analytical solution (Cottrell [181], see also standard texts such as Bard and Faulkner [73, 74] or Galus [257]):

$$c(x,t) = c^* \operatorname{erf}\left(\frac{x}{2\sqrt{Dt}}\right) \ . \tag{2.35}$$

In electrochemical experiments, we usually want the current or, since it is related simply by (2.9) to $\partial c/\partial x$ at $x = 0$, we want $(\partial c/\partial x)_0$. This is obtained by differentiating (2.35) and setting $x = 0$, resulting in

$$\left(\frac{\partial c}{\partial x}\right)_0 = \frac{c^*}{\sqrt{\pi D t}} \tag{2.36}$$

and the current itself is given by

$$i = \frac{nFA\sqrt{D}c^*}{\sqrt{\pi t}}\,, \tag{2.37}$$

the **Cottrell equation**.

The function erf is the error function, for which tables exist [28], and which can be numerically computed (see the function `ERF` in the examples). The solution, (2.35), is shown in Fig. 2.4 for three values of t, increasing as the curves go to the right.

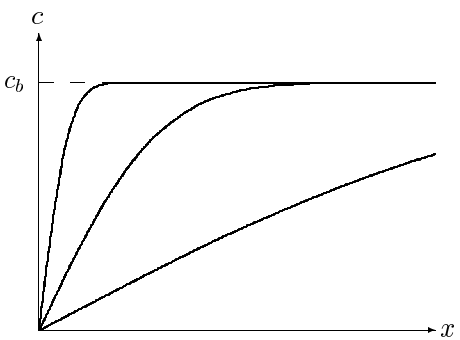


Fig. 2.4. Concentration profile changing with time for the Cottrell experiment

These so-called concentration profiles agree with our intuitive picture of what should happen. Note that the concentration gradient at $x = 0$ decreases with time. The current function declines with the inverse square root of time (2.36). If, for a particular t value, we wish to know the current, we can insert c^*, D and t into this equation and use (2.9) to get it.

It is clear from Fig. 2.4 that we should be able to define a distance that roughly corresponds, at a given time, to the distance over which much of the concentration change has taken place. One possible choice for this is the distance δ as shown in Fig. 2.5, obtained by continuing the concentration gradient at $x = 0$ straight up to c^*. Since this tangent line has the equation

$$c = \left(\frac{\partial c}{\partial x}\right)_0 x = \frac{c^* x}{\sqrt{\pi D t}}\,, \tag{2.38}$$

δ will be obtained by substituting $c = c^*$ and $x = \delta$; this leads to

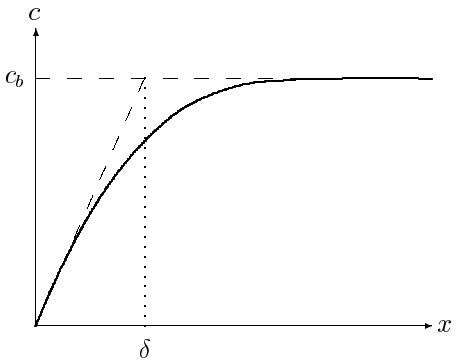


Fig. 2.5. The diffusion layer thickness δ

$$\delta = \sqrt{\pi D \tau} \tag{2.39}$$

now expressed for the particular observation time τ. This quantity – a length scale – was defined by Nernst (and Brunner) in 1904 [158,410], and is named after the former. We find that, at any given time, there will be noticeable concentration changes in the solution within a space extending only a few multiples of δ.

This definition of δ is one of several possible. The way it is defined above yields that distance for which the concentration has moved from zero to c^* by a fraction $\mathrm{erf}(\frac{1}{2}\sqrt{\pi}) \approx 0.8$ or in other words, about 80% of the change has happened at that point. Although this might be the most rational definition, others can be agreed upon. In the present context, it turns out that a smaller distance is the most convenient:

$$\delta = \sqrt{D\tau}\ . \tag{2.40}$$

At this distance, about 52% of the total change has happened. This definition of δ will be used in the remainder of the book.

This scale is now used. The three variables c, x and t are rendered dimensionless by the normalisations in (2.25) and applying these to (2.33) results in the new dimensionless diffusion equation

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} \tag{2.41}$$

and for the Cottrell system in these terms, the dimensionless boundary conditions,

$$\begin{aligned} T < 0, \text{all } X: &\quad C = 1\ , \\ T \geq 0, X = 0: &\quad C = 0\ , \\ \text{all } T, X \to \infty: &\quad C = 1\ . \end{aligned} \tag{2.42}$$

From both the diffusion equation and the boundary conditions, such parameters as D and c^* have now been eliminated. The solution is then

$$C(X,T) = \mathrm{erf}\left(\frac{X}{2\sqrt{T}}\right) \tag{2.43}$$

for the concentrations and

$$\left(\frac{\partial C}{\partial X}\right)_0 = G = \frac{1}{\sqrt{\pi T}} \,. \tag{2.44}$$

This might be called the dimensionless Cottrell equation, for "current" G, which in fact is the dimensionless concentration gradient at $X = 0$.

Potential Step, Reversible System

In the Cottrell experiment, as described in the last section, we have a step to a very negative potential, so that the concentration at the electrode is kept at zero throughout. It is possible also to step to a less extreme potential. If the system is reversible, and we consider the two species A and B, reacting as in (2.18), then we have the Nernstian boundary condition as in (2.24). Using (2.29) and assigning the symbols C_A and C_B, respectively, to the dimensionless concentrations of species A and B, we now have the new boundary conditions for the potential step,

$$\begin{aligned} T < 0, \text{all } X: &\quad C_A = 1, C_B = 0 \,, \\ T \geq 0, X = 0: &\quad C_A/C_B = e^p \,, \\ \text{all } T, X \to \infty: &\quad C_A = 1, C_B = 0 \,, \end{aligned} \tag{2.45}$$

in which species B is not initially present. Note that substance A is now the reference species and the values of its diffusion coefficient D_A and its initial bulk concentration c_A^* are the ones used in the normalisations (2.25) and (2.40). Similarly, if the diffusion coefficients are different for the two species, we also define the ratio

$$d = D_B/D_A \,. \tag{2.46}$$

There is now the additional boundary condition (flux condition),

$$f_A + f_B = 0 \tag{2.47}$$

or, in terms of concentration gradients at the electrode,

$$D_A \left.\frac{\partial c_A}{\partial x}\right|_{x=0} + D_B \left.\frac{\partial c_B}{\partial x}\right|_{x=0} = 0 \tag{2.48}$$

which, in its dimensionless form and using (2.46), becomes

$$\left.\frac{\partial C_A}{\partial X}\right|_{X=0} + d \left.\frac{\partial C_B}{\partial X}\right|_{X=0} = 0 \,. \tag{2.49}$$

The solution to all this is, as given in Galus [257], is

$$C_A(X,T) = \frac{d^{-1/2}e^p + \operatorname{erf}\left(\frac{X}{2\sqrt{T}}\right)}{1 + d^{-1}e^p} \tag{2.50}$$

and for C_B

$$C_B(X,T) = \frac{d^{-1/2}\operatorname{erfc}\left(\frac{X}{2\sqrt{dT}}\right)}{1 + d^{-1}e^p} \tag{2.51}$$

and the current (expressed as the dimensionless gradient for A) is

$$G = G_{Cott}/(1 + d^{-1}e^p) \tag{2.52}$$

where G_{Cott} is the G-value for the simple Cottrell case as in (2.44).

If the two species' diffusion coefficients are assumed equal ($d = 1$), the above equations simplify in an obvious way. In fact, then the problem is mathematically equivalent to the simple Cottrell case. Cottrell pointed out [181] that then, initially the concentrations at the electrode of the two species will instantly change to their Nernstian values and remain there after that.

A final point concerns the fact that, if indeed $d = 1$, then at any point X,

$$C_A(X,T) + C_B(X,T) = C_A(X,0) + C_B(X,0)\ . \tag{2.53}$$

This equation could be used to simplify the simulation, reducing it to only a single species to be simulated. Agreeing with Feldberg however (private communication), this is not a good idea. Rather, the above equation should be used as a check on a given simulation, to make sure that all is well.

Potential Step, Quasi- and Irreversible System

For the **quasireversible** case, two species A and B must again be considered and the two boundary conditions are the flux condition (2.49) and the dimensionlesss form of the Butler-Volmer equation. The forward and backward heterogeneous rate constants k_f and k_b are normalised:

$$K_f = k_f\sqrt{\frac{\tau}{D_A}} \tag{2.54}$$

$$K_b = k_b\sqrt{\frac{\tau}{D_A}} \tag{2.55}$$

and the dimensionless current G is as given in (2.30). With suitable discretisation, G becomes one of the two boundary conditions, the other one being the usual flux expression (2.47).

This case was studied and published in 1952–3 by several groups independently, some giving the solution for the case of both K_f and K_b being nonzero and some treating the totally irreversible case, $K_b = 0$. See [257] for the references. Texts tend to give only the solution for the current, but

by continuing the treatments in [257] (p. 235) or [73, 74], the solution, in dimensionless form, is

$$C_A(X,T) = 1 - \frac{K_f}{H^*}\left\{-\exp(H^*X)\exp(H^{*2}T)\operatorname{erfc}\left(H^*\sqrt{T} + \frac{X}{2\sqrt{T}}\right) + \operatorname{erfc}\left(\frac{X}{2\sqrt{T}}\right)\right\} \tag{2.56}$$

for the concentration profile of species A and

$$C_B(X,T) = \frac{K_f}{H^*}\left\{-\exp(H^*X)\exp(H^{*2}T)\operatorname{erfc}\left(H^*\sqrt{T} + \frac{X}{2\sqrt{dT}}\right) + \operatorname{erfc}\left(\frac{X}{2\sqrt{dT}}\right)\right\} \tag{2.57}$$

for species B, where the dimensionless variable H^* is defined as

$$H^* = H\sqrt{\tau} \tag{2.58}$$

using the symbol H as defined in [73, 74],

$$H = \frac{k_f}{\sqrt{D_A}} + \frac{k_b}{\sqrt{D_B}}\ . \tag{2.59}$$

The dimensionless current then is

$$G = K_f \exp(H^{*2}T)\operatorname{erfc}(H^*\sqrt{T})\ . \tag{2.60}$$

Modification to the totally irreversible case ($k_b = 0$) is trivial, as is the simplification to equal diffusion coefficients ($d = 1$).

Potential Step, Homogeneous Chemical Reactions

Three examples are popular here. The first two start with flash photolysis, where an intense flash irradiates the whole cell at $t = 0$, instantly producing an electrochemically active species that decays chemically in time, either by a first-order reaction, or a second-order reaction. The labile substance is assumed to be formed uniformly in the cell space with a bulk concentration of c^*. These are cases where the concentration at the outer boundary is not constant, falling with time. The third case, the catalytic or **EC′** system (see [73, 74]), is of special interest because of the reaction layer it gives rise to.

The **Reinert-Berg** system is the one in which the reactions are

$$\begin{aligned} A + e^- &\rightarrow B \\ A &\rightarrow prod\ , \end{aligned} \tag{2.61}$$

the two reactions taking place simultaneously. We need only consider the single species A. This system poses no special problems. Reinert and Berg solved it [464] for a potential step to very negative potentials, that is, doing a Cottrell experiment on this system. The diffusion equation becomes

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} - kc \tag{2.62}$$

where k is the rate constant of the homogeneous chemical reaction and boundary conditions are

$$\begin{aligned} t = 0\,,\ \text{all } x: &\quad c = c^* \\ t \geq 0\,,\ x = 0: &\quad c = 0 \\ \text{all } t, x \to \infty: &\quad c = c^* e^{-kt}\ . \end{aligned} \tag{2.63}$$

Note the difference from (2.34). Normalising as usual, with the additional normalisation of rate constant k to K:

$$K = k\tau \tag{2.64}$$

so that these equations become

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} - KC \tag{2.65}$$

and

$$\begin{aligned} T = 0\,,\ \text{all } X: &\quad C = 1 \\ T \geq 0\,,\ X = 0: &\quad C = 0 \\ \text{all } T, X \to \infty: &\quad C = e^{-KT}\ . \end{aligned} \tag{2.66}$$

The solution of this is the one for the simple Cottrell system, multiplied by the decay factor

$$C(X,T) = \exp(-KT)\,\mathrm{erf}\left(\frac{X}{2\sqrt{T}}\right) \tag{2.67}$$

and

$$G(T) = \exp(-KT)\frac{1}{\sqrt{\pi T}}\ . \tag{2.68}$$

Obviously, one must choose the characteristic (observation) time τ reasonably – several multiples of the half-life – so that even out in the bulk, there is still some substance left at that time, or else the calculation will be operating on values very close to zero. This will depend on the value of K. When simulating the plain Cottrell experiment, it is customary to simulate to $T = 1$, but here, one might only go to $T = n/K$, with n some smallish number, so that $\exp(-KT)$ does not become too small.

In the Reinert-Berg system, the homogeneous chemical reaction involves a bulk species, and there is no reaction layer (Sect. 2.2.6).

The **Birk-Perone** system, a flash photolysis experiment with subsequent second-order decay, is a little more interesting because it can, with an unsuitable simulation method, lead to negative concentration values. The simultaneous reactions are

$$\begin{aligned} A + e^- &\rightarrow B \\ 2A &\rightarrow prod \end{aligned} \tag{2.69}$$

and this has the governing equation

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} - 2kc^2 \ , \tag{2.70}$$

where k is the rate constant of the homogeneous chemical reaction and boundary conditions are

$$\begin{aligned} t = 0 \, , \ \text{all } x : &\quad c = c^* \\ t \geq 0 \, , \ x = 0 : &\quad c = 0 \\ \text{all } t, x \rightarrow \infty : &\quad c = c^*/(1 + 2ktc^*) \ . \end{aligned} \tag{2.71}$$

The boundary condition at $X \rightarrow \infty$ is the solution of the simple homogeneous reaction taking place there. Normalising all variables, and k normalised using

$$K = 2kc^*\tau \ , \tag{2.72}$$

these become

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} - KC^2 \ , \tag{2.73}$$

with

$$\begin{aligned} T = 0 \, , \ \text{all } X : &\quad C = 1 \\ T \geq 0 \, , \ X = 0 : &\quad C = 0 \\ \text{all } T, X \rightarrow \infty : &\quad C = (1 + KT)^{-1} \ . \end{aligned} \tag{2.74}$$

A solution for this system was first attempted by Birk and Perone [121], who however oversimplified their assumptions. This was pointed out later [146] and the more rigorous solution (current only) was found to be

$$\frac{i_k}{i_{k=0}} = \frac{1}{1+\theta}\left\{1 + \sum_{n=1}^{\infty} a_n \left(\frac{\theta}{1+\theta}\right)^n\right\} \ , \tag{2.75}$$

in which $i_{k=0}$ is the plain Cottrell solution, θ is defined as $2kc^*t = KT$ and the first 10 coefficients a_n are [146]

$$
\begin{aligned}
a_1 &= 4/\pi - 1 = 0.27324 \\
a_2 &= 0.08327 \\
a_3 &= 0.02893 \\
a_4 &= 0.01162 \\
a_5 &= 0.00540 \\
a_6 &= 0.00286 \\
a_7 &= 0.00169 \\
a_8 &= 0.00108 \\
a_9 &= 0.00074 \\
a_{10} &= 0.00053 \ .
\end{aligned}
$$

Another system of interest in connection with potential steps (and, see below, LSV) is the **catalytic** or **EC′** system, described in simplified form by

$$
\begin{aligned}
A + e^- &\Leftrightarrow B \\
B &\rightarrow A \ ,
\end{aligned}
\tag{2.76}
$$

where the product B reverts, with pseudo-first-order rate constant k, to the original A. The first reaction is conveniently taken to be diffusion limited (that is, the potential is very negative as in the Cottrell experiment). Normalising as usual (rate constant k as above, (2.64)) and assuming equal diffusion coefficients ($d = 1$), the boundary conditions are

$$
\begin{aligned}
T < 0, \text{all } X: &\quad C_A = 1, C_B = 0 \\
T \geq 0, X = 0: &\quad C_A = 0 \\
\text{all } T, X \rightarrow \infty: &\quad C_A = 1, C_B = 0 \ .
\end{aligned}
\tag{2.77}
$$

The solution, derived by Delahay and Stiehl [202] in dimensionless form:

$$
\begin{aligned}
C_A(X,T) = 1 &- \tfrac{1}{2}\exp(X\sqrt{K})\operatorname{erfc}\left(\frac{X}{2\sqrt{T}} + \sqrt{KT}\right) \\
&- \tfrac{1}{2}\exp(-X\sqrt{K})\operatorname{erfc}\left(\frac{X}{2\sqrt{T}} - \sqrt{KT}\right) \\
C_B(X,T) = &\ \tfrac{1}{2}\exp(X\sqrt{K})\operatorname{erfc}\left(\frac{X}{2\sqrt{T}} + \sqrt{KT}\right) \\
&+ \tfrac{1}{2}\exp(-X\sqrt{K})\operatorname{erfc}\left(\frac{X}{2\sqrt{T}} - \sqrt{KT}\right)
\end{aligned}
\tag{2.78}
$$

and the current

$$
G = \sqrt{K}\operatorname{erf}\sqrt{KT} + \frac{\exp(-KT)}{\sqrt{\pi T}} \ . \tag{2.79}
$$

This system will be discussed again in a later chapter, because it is of special interest, both species A and B forming a reaction layer. The thickness μ of

this layer was given in Sect. 2.2.6 and this can now be normalised, for a first-order homogeneous reaction, for which we have the dimensionless rate constant as in (2.64), giving the dimensionless reaction layer thickness

$$\mu^* = \frac{1}{\sqrt{K}} \,. \tag{2.80}$$

Note that for large K and T, the current G approaches the constant value $\sqrt{K}$.

2.4.2 Constant Current

While the Cottrell system might be regarded as the simplest possible model with a Dirichlet boundary condition (that is, in which boundary concentrations are specified), the constant current case is the simplest possible for the Neumann boundary condition, in which a concentration gradient is specified at the boundary. This model can also be called the chronopotentiometric experiment since here, the current is given and it is the electrode potential that is measured against time. Mathematically this model is defined by the usual (2.33), here with the boundary conditions

$$\begin{aligned} t < 0, \text{all } x: &\quad c = c^* \\ t \geq 0, x = 0: &\quad \mathrm{d}c/\mathrm{d}x = const \,. \\ \text{all } t, x \to \infty: &\quad c = c^* \,. \end{aligned} \tag{2.81}$$

The solution to this, that is, the concentration profile as a function of x and t, is [74]

$$c(x,t) = c^* - \frac{i}{nFAD}\left\{2\sqrt{\frac{Dt}{\pi}}\exp\left(-\frac{x^2}{4Dt}\right) - x\,\mathrm{erfc}\left(\frac{x}{2\sqrt{Dt}}\right)\right\} \,, \tag{2.82}$$

where i is the constant current that is applied, D is the diffusion coefficient of the electroactive species. Some concentration profiles at three time values are shown in Fig. 2.6 and the constant concentration gradient at $x = 0$ can be seen. Also, the concentration $c(0,t)$ decreases with time t; it is in fact

$$c(0,t) = c^* - \frac{2i\sqrt{t}}{nFA\sqrt{\pi D}} \tag{2.83}$$

and reaches zero at some time, as shown in the figure. This time is the transition time (so named because the electrode potential undergoes a sharp transition at this point). It is given the symbol τ and is related to the current i by the **Sand equation**:

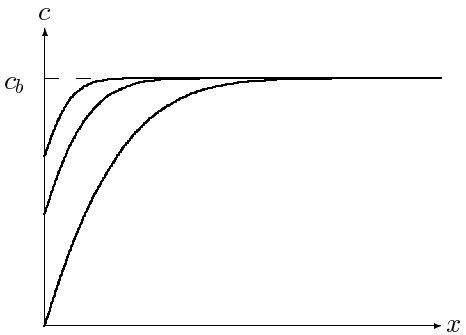


Fig. 2.6. Concentration profile changing with time for chronopotentiometry

$$\frac{i\sqrt{\tau}}{c^*} = \frac{nFA\sqrt{\pi D}}{2} \tag{2.84}$$

first given by Sand [493] and, with more detail, by Karaoglanoff [331].

To normalise this system, the previous definition (2.40) is used for the distance x, and c^*, the bulk concentration, for the concentration c; for the time unit, it is natural to use the transition time τ itself. This makes the boundary conditions

$$\begin{aligned} T < 0, \text{all } X: &\quad C = 1\,, \\ T \geq 0, X = 0\ : &\quad \mathrm{d}C/\mathrm{d}X = \tfrac{1}{2}\sqrt{\pi} \\ \text{all}, X \to \infty\ T: &\quad C = 1\,. \end{aligned} \tag{2.85}$$

Interestingly, here the constant current becomes the dimensionless constant concentration gradient at the electrode, with the value $\frac{1}{2}\sqrt{\pi}$. The dimensionless concentration profile is

$$C(X,T) = 1 - \sqrt{T}\left\{\exp\left(-\frac{X^2}{4T}\right) - \frac{\sqrt{\pi}X}{2\sqrt{T}}\,\mathrm{erfc}\left(\frac{X}{2\sqrt{T}}\right)\right\} \tag{2.86}$$

and, very simply,

$$C(0,T) = 1 - \sqrt{T}\,. \tag{2.87}$$

In Fig. 2.6, the profiles shown are for $t = 0.1\tau$, $t = 0.3\tau$ and $t = \tau$; that is, for $T = 0.1$, $T = 0.3$ and $T = 1$.

2.4.3 Linear Sweep Voltammetry (LSV)

This is another useful system with which methods can be tested, one reason being that it demands more iterations than those mentioned above and is thus notoriously time-consuming. We again consider the simple reaction

$$A + ne^- \Leftrightarrow B \tag{2.88}$$

and assume reversibility. The electrode potential $E(t)$ is time-dependent,

$$E(t) = E_1 + vt \tag{2.89}$$

in which E_1 is the starting potential and v is the scan rate in $\mathrm{V\,s^{-1}}$. The diffusion equations are as for the potential step with a reversible system (2.18) with the boundary conditions, for the classical case,

$$\begin{aligned} t < 0: &\quad E = E_1 \\ t < 0, \text{all } x: &\quad c_A = c^*, c_B = 0 \\ t \geq 0: &\quad E(t) = E_1 + vt \\ t \geq 0, x = 0: &\quad c_A/c_B = \exp\left\{\tfrac{n\mathcal{F}}{\mathcal{RT}}\left(E(t) - E^0\right)\right\} \\ t \geq 0, x \to \infty: &\quad c_A = c^*, c_B = 0\,, \end{aligned} \tag{2.90}$$

where c^* is the initial bulk concentration of species A and species B is not present initially. A common diffusion coefficient for both species, D, is assumed. In practice, the sweep terminates at some (more negative) potential E_2, but this is not part of the description. This system is interesting in that it was in fact the first to be simulated, by Randles, in 1948 [460] using hand calculations. In the same year, Ševčík [505] worked towards an analytical solution, ending in an integral equation he was forced to solve numerically. The current function is therefore called the Randles-Ševčík function. The integral equation was developed in 1964 by Nicholson and Shain [417] and solved numerically with greater accuracy. Their calculations were later improved by Oldham [426], Mocak [400] and Mocak & Bond [401] who used series solutions. The Oldham values have not been improved upon. The current function (which will be seen below to be the dimensionless flux for species A at the electrode), given the symbol χ, was found by Oldham to have a peak value at the dimensionless potential p_{max} (for the definition see (2.29)) of -1.1090, corresponding to $-28.493/n\,\mathrm{mV}$ (at 25^oC and using the Diehl value [211] for the Faraday, $96486.0\,\mathrm{C/mol}$), the peak χ (or G) value there being 0.44629. These numbers are useful to know as standards for comparing simulations, and refer only to the LSV case (that is, no reverse sweep is described).

To render the LSV system dimensionless, the usual reference values for concentration, time and distance from the electrode are needed, as well as that for potential (2.29) (and thus, sweep rate). Both species' concentrations are normalised by the initial bulk concentration of A, c^*, as always, and the potential to dimensionless p as in (2.29), (2.89) thus becoming

$$p = p_1 - at \tag{2.91}$$

with p_1 being the dimensionless starting potential, and the variable a given by

$$a = \frac{n\mathcal{F}}{\mathcal{RT}}\,v\,, \tag{2.92}$$

the dimensionless sweep rate (and now sweeping in the cathodic direction). A reference time τ can now be defined, being simply the time it takes to sweep through one p-unit,

$$\tau = a^{-1} \tag{2.93}$$

so that we have, as usual,

$$T = t/\tau \tag{2.94}$$

and also the reference distance δ, as before,

$$\delta = \sqrt{D\tau} \tag{2.95}$$

and thus

$$X = x/\delta \tag{2.96}$$

so that we now have the two diffusion equations

$$\begin{aligned} \frac{\partial C_A}{\partial T} &= \frac{\partial^2 C_A}{\partial X^2} \\ \frac{\partial C_B}{\partial T} &= \frac{\partial^2 C_B}{\partial X^2} \end{aligned} \tag{2.97}$$

with the boundary conditions

$$\begin{aligned} T = 0: &\quad p = p_1 \\ T \le 0, \text{ all } X: &\quad C_A = 1, C_B = 0 \\ T \ge 0: &\quad p(T) = p_1 - T \\ T \ge 0,\ X = 0: &\quad C_A/C_B = \exp(p(T)) \\ T \ge 0,\ X \to \infty: &\quad C_A = 1, C_B = 0\,. \end{aligned} \tag{2.98}$$

Note the rather simple form of the Nernst equation here, and the fact that the dimensionless sweep rate is now unity, that is, one p-unit per one T-unit.

When solving this system by computer, the dimensionless results can then be translated back into dimensioned values. The above χ function is the same as dimensionless $\partial C_A/\partial X\,(X = 0)$ or G, and becomes a real current via the equation

$$i(t) = nFADG = nFAD\chi \tag{2.99}$$

at the actual potential

$$E(t) = \frac{\mathcal{RT}}{n\mathcal{F}}\, p(t) + E^0\,. \tag{2.100}$$

With LSV, the quasireversible and irreversible cases might also be interesting models, both of which have mixed boundary conditions, lying somewhere between the extremes of Dirichlet and Neumann conditions, because here we have fluxes at the electrode, determined by heterogeneous rate constants (depending on potential) and concentrations at the electrode. Also,

as will be seen in a later chapter, these models give rise, with some simulation methods, to surprising instabilities. These models are described in the standard texts such as [73, 74] and [257].

The quasireversible LSV case was treated by Matsuda and Ayabe [389], who used a series sum as an approximation to the integral equation obtained from the Laplace-transform solution of the problem. The result depends on the heterogeneous rate constant, both the peak current and the peak potential varying with this parameter. Basha et al. [82] tried to improve on the results but it seems that those of Nicholson and Shain [417] were better. These also provided results for the totally irreversible case, first described by Delahay [199]. For this, the $\chi(at)$-function has a constant maximum, given to four figures in [73, 74], 0.4958, from the tables in [417]. Peak potential varies with rate constant, as with the quasireversible case.

Thus, for these two cases, we do not have high-precision comparisons. In such a case, one recourse is to do convergence computations, going to finer and finer intervals until there is no further change in the values, which can then be used for reference. This of course rests on the assumption that one has a method that is guaranteed to work; and often, this is reasonable. One uses a tried-and-true method with such a guarantee but which is not necessarily highly efficient. The values are then used to check whatever method one is working on, that might be more efficient. If this strikes the reader as somewhat unsatisfactory, note that, for LSV, there are in fact no analytical solutions at all, except those based on numerical methods of some kind, by either solving the associated integral equation numerically, or evaluating various series sums that are considered good approximations to the solution of such integral equations. It can be considered that digital simulation is one further method for doing this job.

Another case of interest with LSV is the catalytic system, described above in the section on potential steps. The equations are the same except that the potential here is not constant and very negative, following (2.29) in its dimensionless form. For small and intermediate rates of the homogeneous chemical reaction (dimensionless constant K), the same procedure as mentioned above, that is, convergence simulations, must be used. For large K, however, the LSV curves become sigmoid, with a plateau equal to the current for the potential step, $G = \sqrt{K}$. This can be used to test methods. Figure 2.7 shows LSV curves for some K-values, where this effect is seen.

2.5 Adsorption Kinetics

Adsorption is often present in electrochemical systems, both unintended and intended for certain purposes. The rate at which adsorbed layers form is of interest and must be simulated in most cases, as it is mathematically difficult, although a few simple cases have been solved. Modern texts [74, 257] give adsorption kinetics quite brief treatment, and the classic literature is rather

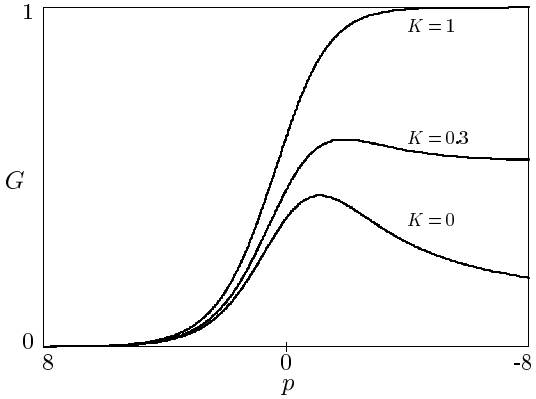


Fig. 2.7. LSV curves, catalytic system

old. Textbooks devoted to the subject (Damaskin et al. [191] or Jehring [315]) deal very briefly with the kinetics of adsorption. The recent focus is on self-assembled monolayers (SAMs; for a review, see for example [243]), and the kinetics of their formation are complicated by changes that take place after adsorption [74]. It used to be thought that the adsorption step itself is fast on mercury, but slower on solid metals [257]. The insight gained from the study of SAMs suggests that on solid metals, too, adsorption as such is fast, but the rearrangement that takes place afterwards, is a slow process [243].

There are some important general relations for a substance adsorbed from solution on an electrode. These pertain to the equilibrium state and the kinetics of the process leading to equilibrium. Adsorption kinetics receives rather intermittent attention in the electrochemical literature. One of the clearest discussions is by Mohilner [403]; see also Delahay [200], Bard and Faulkner [74].

The degree of adsorption is expressed either by Γ, the surface concentration, in units of moles per unit area, or in terms of the fractional coverage θ:

$$\theta = \Gamma/\Gamma_m \ , \tag{2.101}$$

where Γ_m is the maximum possible surface concentration at saturation, in many cases corresponding to a complete monolayer of the substance on the electrode. At equilibrium, Γ or θ are related to the adsorbed substance's concentration c_0 adjacent to the electrode, by the adsorption isotherm I, customarily written in the inverse form

$$bc_0 = I(\theta) \tag{2.102}$$

or, in dimensionless terms,

$$BC_0 = I(\theta) \tag{2.103}$$

with $B = bc^*$. It may be that while the state of adsorption is in momentary equilibrium with c_0, there are nevertheless concentration gradients in

the solution. The isotherms take many forms; a large list was presented by Mohilner [403]. A few examples are

$$I(\theta) = \theta \tag{2.104}$$

which is the linear or Henry isotherm, sometimes applicable for $\theta \ll 1$; or the Langmuir isotherm

$$I(\theta) = \frac{\theta}{1-\theta} \; . \tag{2.105}$$

If there is interaction between the particles (attractive or repulsive), the Frumkin isotherm may apply:

$$I(\theta) = \frac{\theta}{1-\theta} \exp(-2a\theta) \tag{2.106}$$

with a the attraction parameter. The logarithmic Temkin isotherm [403] is

$$I(\theta) = \exp(a\theta) \; . \tag{2.107}$$

There are other, more complicated isotherms but the above examples suffice.

In order to reach a certain surface concentration Γ or fractional coverage θ, the substance in question must first arrive at the electrode by some transport process. The diffusion equation applies for this part. The rate of increase of Γ (per unit area) is proportional to the unit area flux at the electrode

$$\frac{d\Gamma}{dt} = -f \tag{2.108}$$

(flux being regarded as away from the electrode). From (2.2) and our definition of g this becomes

$$\frac{d\Gamma}{dt} = Dg \; . \tag{2.109}$$

We prefer to work with θ and so get (using (2.101))

$$\frac{d\theta}{dt} = Dg/\Gamma_m \; . \tag{2.110}$$

In dimensionless form, using the transformations (2.25) as before, this is

$$\frac{d\theta}{dT} = \frac{c^* \sqrt{D\tau}}{\Gamma_m} G \; , \tag{2.111}$$

τ being the observation time as before. Some model systems with solutions exist for adsorption kinetics. The simplest case is that of a very large B parameter in (2.103), and a Cottrellian experiment. For $B \to \infty$, there is then the set of boundary conditions

$$\begin{aligned} T<0: \quad & C(X,T)=1; \quad \theta=0 \\ T\geq 0: \quad & C(0,T)=0; \quad \theta=\tfrac{c^*\sqrt{D\tau}}{\Gamma_m}\int_0^T G dT \\ \text{all } T, X\to\infty: \quad & C=1\ . \end{aligned} \tag{2.112}$$

In other words, adsorption is so strong, that every particle of the substance arriving at the electrode is adsorbed immediately, forcing the concentration there, in solution, to zero. So the adsorbate simply accumulates by the Cottrellian flux given by the potential step experiment, and the solution, given first by Koryta [342], is

$$\theta(T)=\frac{2c_b\sqrt{D\tau}}{\Gamma_m\sqrt{\pi}}\sqrt{T}\ . \tag{2.113}$$

If adsorption is fast but not sufficiently strong to justify the assumption $C_0\approx 0$, then C_0 will, at any instant, be determined by the adsorption isotherm (2.103). This boundary condition leads to mathematical problems; the integral equation resulting from (2.110) then becomes a Volterra equation. This has been solved for only some very simple isotherms. Delahay and Trachtenberg [203] solved it for the Henry isotherm (2.104), the solution being

$$C=1-\exp\left[\frac{X}{K}+\frac{T}{K^2}\right]\operatorname{erfc}\left[\frac{X}{2\sqrt{T}}+\frac{\sqrt{T}}{K}\right]\ , \tag{2.114}$$

in which the dimensionless $K=b\Gamma_m/\sqrt{D\tau}$.

Reinmuth [465] arrived at a solution for the Langmuir isotherm in the form of a series, involving the beta function. Levich et al. [362] had an approximate solution for a general adsorption isotherm.

If the adsorption step itself is rate-limiting, one must have available rate expressions for the adsorption and the desorption steps. The flux in (2.108) is then split into two opposing components. Using the notation of Delahay and Mohilner [201,403], there is a forward flux v_f, adding to the adsorbate's surface concentration and backward flux v_b, the rate of redissolution of adsorbed substance. These obey rate equations rather analogous to those for electron transfer, the Butler-Volmer equation, in the sense that there are rate constants that are potential dependent. For the forward and backward rates, we have

$$\begin{aligned} v_f &= k_f f_1(c_0,\Gamma) \\ v_b &= k_b f_2(\Gamma)\ , \end{aligned} \tag{2.115}$$

where k_f and k_b are the forward and backward rate constants and f_1 and f_2 are functions whose forms depend on the adsorption isotherm assumed. Note that the two constants have different units. If the forward and backward rates are equal, then there is equilibrium and the isotherm is obtained by equating

the right-hand sides of (2.115). If the rates are not equal, there will be a net flux of substance between the two phases.

Various workers [201,370,371,403] have presented particular forms of the functions f_1 and f_2, for various isotherms. For all these equations, one needs to know the k parameters and possibly more. In all cases, we have

$$\frac{d\Gamma}{dt} = v_f - v_b \tag{2.116}$$

which we would usually normalise to the dimensionless form

$$\frac{d\theta}{dT} = V_f - V_b \tag{2.117}$$

with $V = v\tau/\Gamma_m$, τ being, as before, a chosen experimental time scale.

Two examples are given here, to be followed up in Chap. 10. Lovrić and Komorsky-Lovrić [371] expressed (2.115) for the simple Henry isotherm (2.104), as

$$\begin{aligned} v_f &= k_f c_0 \\ v_b &= k_b \Gamma \,, \end{aligned} \tag{2.118}$$

which is linear in both c_0 and Γ, while Lorenz [370] expressed the equation for the Langmuir isotherm (2.105) as

$$\begin{aligned} v_f &= k_f c_0 (\Gamma_m - \Gamma) \\ v_b &= k_b \Gamma \end{aligned} \tag{2.119}$$

where we now have a nonlinear term in the first equation. These can be normalised conveniently. The v are normalised to V as above. Letting $K_f = k_f \tau c^*/\Gamma_m$ and $K_b = k_b \tau c^*$ (note the different normalisations of the two rate constants), we obtain

$$\begin{aligned} V_f &= K_f C_0 \\ V_B &= K_b \theta \end{aligned} \tag{2.120}$$

for the Henry isotherm, and

$$\begin{aligned} V_f &= K_f C_0 (1 - \theta) \\ V_b &= K_b \theta \end{aligned} \tag{2.121}$$

for the Langmuir isotherm. These two cases – one linear and the other nonlinear – are good examples for the simulation process. More complex isotherms such as the Frumkin isotherm also lead to nonlinear equations, so that this one nonlinear example suffices to point the way. How all this is simulated is described in Chap. 10.

3 Approximations to Derivatives

In this chapter, all the discrete approximations required for simulation are established, that is, for first and second derivatives, both central and asymmetric forms, and for a range of numbers of points used.

3.1 Approximation Order

Consider Fig. 3.1 and assume that the marked points on the x-axis are equidistant with length h between them. For the moment, consider a simple first derivative "around the region" between x_1 and x_2 (to be made more precise in later sections). The function shown as a line is known only in the form of the fat points on it. Intuitively, one thinks of the approximation in that region as

$$\frac{dy}{dx} = \frac{y_2 - y_1}{h} \tag{3.1}$$

and this can be used to define the concept of *order*. The above expression is that for the slope of the straight line drawn from the first point (at x_1) to the second (at x_2), and is slightly in error. The error depends on where on the curve we mean the expression to apply. In later sections all this will be developed precisely, but here it suffices to say that the error will usually be a function of the interval h, and will become smaller, as we make h smaller. The order then tells us how much smaller. The relation between the error e and h is approximately given by

$$e = const \times h^p \tag{3.2}$$

and the power p is the order (while the constant is of lesser importance). For example, it is seen below that, if we mean (3.1) to apply at the position x_1 or x_2, then p is equal to unity, and one says that the approximation is first-order with respect to h, expressed as $O(h)$. In other words, if we attempt to make the error smaller by halving h, then we also halve the error. If we mean the approximation to apply to the point midway between the interval, that is, at $x = (x_1 + x_2)/2$, it turns out that $p = 2$, and the error is $O(h^2)$. Thus, if we halve h, the error becomes a quarter as large. This is better than first-order and generally, one seeks approximations of high order.

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 33–49 (2005)
www.springerlink.com

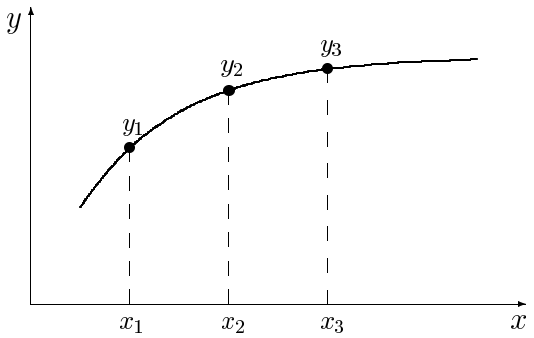


Fig. 3.1. Arbitrary function with 3 points marked

In digital simulation, when discretising the diffusion equation, we have a first derivative with respect to time, and one or more second derivatives with respect to the space coordinates; sometimes also spatial first derivatives. Efficient simulation methods will always strive to maximise the orders.

3.2 Two-Point First Derivative Approximations

Consider again Fig. 3.1, and the point at x_2, expressed as a Taylor series development going from x_1. Note that the symbols y_2, $f(x_2)$ and $f(x_1 + h)$ are all synonymous. The Taylor expansion is

$$y_2 = y_1 + hy_1' + \frac{h^2}{2!}y_1'' + \frac{h^3}{3!}y_1''' + \dots \tag{3.3}$$

where y_1' etc are the progessively higher spatial derivatives of the function at the point (x_1, y_1). This equation can be rearranged to

$$y_1' = \frac{y_2 - y_1}{h} - \frac{h}{2!}y_1'' - \frac{h^2}{3!}y_1''' - \dots \tag{3.4}$$

where the first term on the right-hand side is in fact (3.1) above. However, now we know more about this approximation: we note that it refers to the point (x_1, y_1), and that if we write

$$y_1' = y'(x_1) = \frac{y_2 - y_1}{h} \tag{3.5}$$

then this has an error equal to the sum of the further terms on the right-hand side of (3.4). That is, the error is

$$e = -\frac{h}{2!}y_1'' - \frac{h^2}{3!}y_1''' - \dots \ . \tag{3.6}$$

This is a polynomial in h, and since h is normally rather small, the lowest power in h will contribute most to the sum. Thus we see that the error is $O(h)$ (the actual coefficients do not matter as much as the order).

The approximation (3.5) is called a *forward difference* because the values used to approximate it lie forward of the point (x_1) where it is meant to apply.

It is possible to develop the point at x_1 going backward from that at x_2, again using the Taylor expansion:

$$y_1 = y_2 - hy_2' + \frac{h^2}{2!}y_2'' - \frac{h^3}{3!}y_2''' + \dots \tag{3.7}$$

where there is now an alternation of sign because of the negative value $-h$. Rearranging this yields

$$y_2' = \frac{y_2 - y_1}{h} + \frac{h}{2!}y_2'' - \frac{h^2}{3!}y_2''' + \dots \tag{3.8}$$

giving the approximation

$$y_2' = y'(x_2) = \frac{y_2 - y_1}{h} \tag{3.9}$$

which is also $O(h)$ (with different polynomial coefficients) and, as it refers to the point (x_2, y_2) using a preceding other point, is a *backward difference*.

It will now become clear why Fig. 3.1 has three points on it. We focus on the point at x_2 and use Taylor's expansions around this point for both y_1 (see 3.7) and y_3:

$$y_3 = y_2 + hy_2' + \frac{h^2}{2!}y_2'' + \frac{h^3}{3!}y_2''' + \dots \ . \tag{3.10}$$

Subtracting (3.7) from (3.10) and rearranging yields

$$y_2' = \frac{y_3 - y_1}{2h} + \frac{h^2}{3!}y_2''' + \dots \tag{3.11}$$

that is,

$$y_2' = \frac{y_3 - y_1}{2h} + O(h^2) \tag{3.12}$$

a second-order *central difference* approximation to the first derivative, which is much better than either the forward or backward formulae in the above. Clearly, we could have done this, focussing on a point midway between x_1 and x_2 (let us call it $x_{1.5}$) and Taylor-expanding around it for the two points y_1 and y_2, thus arriving at the approximation, $y_{1.5}$ at $x_{1.5}$,

$$y_{1.5}' = \frac{y_2 - y_1}{h} + O(h^2) \tag{3.13}$$

which is also second-order with respect to h.

We now have three two-point approximations for a first derivative, all in fact being the same expression, $(y_2 - y_1)/h$, but depending on where this formula is intended to apply, being, respectively a forward difference of $O(h)$ if applied at x_1, a backward difference of $O(h)$ if applied at x_2 and a central difference of $O(h^2)$ if applied at $(x_1 + x_2)/2$. In subsequent chapters, all these will be used to approximate, among others (2.3)–(2.8).

3.3 Multi-Point First Derivative Approximations

The above approximations to a first derivative used only two points, which sets a limit on the approximation order. By using more points, higher-order approximations can be achieved. In the context of this book, forward and backward multi-point formulae are of special interest, as well as some asymmetric and central multi-point ones. To this end, a notation will be defined here. Figure 3.2 shows the same curve as Fig. 3.1 but now seven points are marked on it. The notation to be used is as follows. If a derivative is approximated using the n values $y_1 \dots y_n$, lying at the x-values $x_1 \dots x_n$ (intervals h) and applied at the point (x_i, y_i), then it will be denoted as $y'_i(n)$ (for a first derivative) and $y''_i(n)$ (for a second derivative).

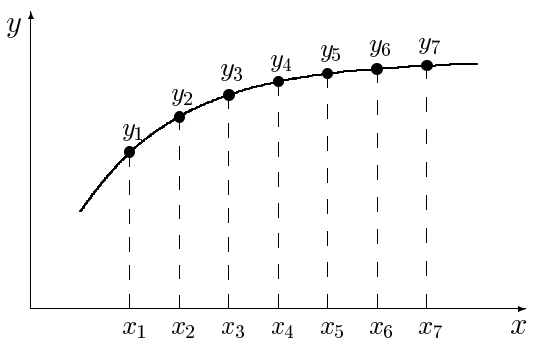


Fig. 3.2. Arbitrary function with 7 points marked

For a given number n of points to be included in an approximation for y' applied at point index i within the group, the procedure is to calculate the β coefficients in the general expression

$$y'_i(n) = \frac{1}{h} \sum_{i=1}^{n} \beta_i y_i \ . \tag{3.14}$$

This is done by writing the Taylor expansions around the point at index i for all the other $(n-1)$ points, to a sufficient number of terms, and solving for the derivatives, discarding all but the numbers for the first derivative.

A single example will illustrate the method. Assume that we want $y_2'(4)$, that is, the derivative y' at point (x_2, y_2) out of points at $x_1 \ldots x_4$ in Fig. 3.2. Taylor expansions are written for points at x_1, x_3, x_4, going to the third derivative (in general, for n points, to the $(n-1)$st derivative):

$$\begin{aligned} y_1 &= y_2 - hy_2' + \frac{h^2 y_2''}{2!} - \frac{h^3 y_2'''}{3!} + O(h^4) \\ y_3 &= y_2 + hy_2' + \frac{h^2 y_2''}{2!} + \frac{h^3 y_2'''}{3!} + O(h^4) \\ y_4 &= y_2 + 2hy_2' + \frac{4h^2 y_2''}{2!} + \frac{8h^3 y_2'''}{3!} + O(h^4)\,. \end{aligned} \tag{3.15}$$

The above can be rewritten as a matrix equation,

$$\begin{bmatrix} -1 & \frac{1}{2!} & -\frac{1}{3!} \\ 1 & \frac{1}{2!} & \frac{1}{3!} \\ 2 & \frac{4}{2!} & \frac{8}{3!} \end{bmatrix} \begin{bmatrix} h & 0 & 0 \\ 0 & h^2 & 0 \\ 0 & 0 & h^3 \end{bmatrix} \begin{bmatrix} y_2' \\ y_2'' \\ y_2''' \end{bmatrix} = \begin{bmatrix} y_1 & -y_2 \\ y_3 & -y_2 \\ y_4 & -y_2 \end{bmatrix} \tag{3.16}$$

(remembering the $O(h^4)$ terms but not writing them). This can be written as

$$\boldsymbol{AHd} = \boldsymbol{b} \tag{3.17}$$

where $\boldsymbol{A}$ is the main matrix, $\boldsymbol{H}$ the diagonal matrix of terms in h, $\boldsymbol{d}$ the solution vector of derivatives $\begin{bmatrix} y_2' & y_2'' & y_2''' \end{bmatrix}^T$ and $\boldsymbol{b}$ the right-hand side vector of knowns. The next step is to multiply by the inverses of the two left-hand matrices

$$\boldsymbol{d} = \boldsymbol{A^{-1}H^{-1}b}\,. \tag{3.18}$$

All that is wanted here is the top row of the inverted matrix, since

$$y_2' = h^{-1} \begin{bmatrix} c_{11} & c_{12} & c_{13} \end{bmatrix} \begin{bmatrix} y_1 & -y_2 \\ y_3 & -y_2 \\ y_4 & -y_2 \end{bmatrix} \tag{3.19}$$

(with c_{11} etc being the first row elements of the inverse $\boldsymbol{C} = \boldsymbol{A}^{-1}$). When inverting matrix $\boldsymbol{A}$, the numbers come out as decimal fractions, in this case $\begin{bmatrix} -\frac{1}{3} & 1 & -\frac{1}{6} \end{bmatrix}$. We prefer whole-number fractions. It is an easy programming job to find a multiplier that makes whole numbers out of all entries in the top row of $\boldsymbol{A}^{-1}$; in this case, it is 6 and the result of the computation is

$$y_2' = \frac{1}{6h} \begin{bmatrix} -2 & 6 & -1 \end{bmatrix} \begin{bmatrix} y_1 & -y_2 \\ y_3 & -y_2 \\ y_4 & -y_2 \end{bmatrix} \tag{3.20}$$

which, when multiplied out and after sorting, gives the result

$$y_2' = \frac{1}{6h}\{-2y_1 - 3y_2 + 6y_3 - y_4\} + O(h^3) \tag{3.21}$$

in which the order term indicates that this approximation is third-order with respect to h.

In this way, the coefficients for any $y_i'(n)$ can be calculated. Table A.1 in Appendix A shows them all, as whole numbers $m\beta_i$, where m is the multiplier mentioned above. For each n, the Table shows forward differences (at index 1), backward derivatives (at index n) and derivatives applying at points between the two ends. For n up to 6, all possible forms are included, as they will be needed later, while for $n = 7$, only the forward and backward formulae are shown, as only these are needed. In case the reader wonders why all this is of interest: the forms $y_1'(n)$ will be used to approximate the current in simulations (see the next section); the backward forms $y_n'(n)$ will be used in the section on the BDF method in Chaps. 4 and 9, and the intermediate forms shown in the Table will be used for the Kimble & White (high-order) start of the BDF method, also described in these chapters. The coefficients have a long history. Collatz [169] derived some of them in 1935 and presents more of them in [170]. Bickley tabulated a number of them in 1941 [88]. The three-point current approximation, essentially $y_1'(3)$ in the present notation, was first used in electrochemistry by Randles [460] (preempted by two years by Eyres et al. [225] for heat flow simulations), then by Heinze et al. [301], and schemes of up to seven-point were provided in [133].

3.4 The Current Approximation

As shown in Chap. 2, (2.26), the current in its dimensionless form G is the dimensionless gradient of C with respect to X at $X = 0$. This implies that a forward difference must be used, as we normally have C-values starting at $X = 0$. There are algorithms with points at negative X values, but they are not generally very successful or popular. The approximation can therefore be expressed as the n-point approximation

$$G \approx \frac{1}{H}\sum_{i=0}^{n-1} \beta_i C_i \; . \tag{3.22}$$

The symbol G_n will sometimes be used, to mark n, the number of points used. The simplest formula is the two-point form,

$$G \approx \frac{1}{H}\,(C_1 - C_0) \tag{3.23}$$

which seems a poor, low-order approximation. It can be justified, however, in cases where H is very small, as is in fact so with most useful programs

these days, since these use unequal intervals, usually spaced very closely near the electrode. As will be seen, this two-point form makes the discretisation of boundary conditions much easier. There are even cases in which the current approximation becomes worse as more points are introduced. This happens with severely stretched grids (see "unequal intervals", elsewhere), so the n-point formula should probably be used only with equal intervals. It has also been argued [100] that the three-point formula,

$$G \approx \frac{1}{2H}\left(-3C_0 + 4C_1 - C_2\right) \tag{3.24}$$

is most compatible with the usual three-point second-order approximation to the second space derivative, being itself second-order. This is a matter of taste.

3.5 The Current Approximation Function $\mathcal{G}$

The above (3.22) is now generalised to operate on any array or vector $\boldsymbol{v} = [v_0, v_1, \ldots, v_{n-1}]^T$, that is, we define the function

$$\mathcal{G}(\boldsymbol{v}, n, H) = \frac{1}{H}\sum_{i=0}^{n-1} \beta_i v_i \tag{3.25}$$

which will be used extensively in this book. Mostly, the second and third arguments will be taken as understood, and the function will then simply be written as $\mathcal{G}(\boldsymbol{v})$ for the general vector $\boldsymbol{v}$, which in many cases will be concentration C (but, as will be seen in Chap. 6, not always).

3.6 High-Order Compact (Hermitian) Current Approximation

There is a trick by which one can increase the order, and thereby the accuracy, of current approximations for a given number of points used. It is related mathematically to the Numerov method, to be discussed in a later chapter. The device is based on the particular form of what we are trying to solve for. It was introduced to electrochemistry by Bieniasz [108,110], referring to some earlier work in other fields. The device belongs to a class of schemes given various names, among them "compact stencil" or "high-order compact (HOC) scheme", or Hermitian. The latter term is possibly the best. It refers to Hermite's interpolation method, clearly described by Kopal [341]. Its essence is the use in an approximation, not only of function values at grid points but also function derivatives. This is now generally applied in other contexts outside interpolation. The English translation of Collatz' book [170] uses the

term to translate the original "Mehrstellenverfahren" and notes that this does not imply that Hermite used the method in this way. The Hermitian method can not only be used to obtain better current approximations, but also for simulations with derivative boundary conditions, to be described in Chap. 9. Another example of a Hermitian method is the Numerov method [421], also described in Chap. 9.

The information on derivatives that the device makes use of is the *pde* itself, which can be written in the form

$$\frac{\partial^2 C}{\partial X^2} = F(X, T, C, \partial C/\partial T) \tag{3.26}$$

writing it out for a normalised equation for simplicity (Bieniasz makes it very general, as a system of such equations, each one with its own diffusion coefficient). The F term always contains the time derivative, but may also contain, for example, homogeneous chemical terms in concentration. In what follows, the function will be simply written as F_i, where it is understood that this refers to the point at $X = iH$.

First we consider the current approximation presented in the above two sections. A question left untouched, for example, the equation for the current approximation (3.25) above, is just what terms were dropped when generating a particular form. The order of what was dropped is given in Sect. 3.3, but not extended to actual higher terms. This must be done now. Bieniasz [108] presents a table of these and we can write the first few of these. For this, it is convenient to use a more compact notation for the higher derivatives: let

$$D_x^k \equiv \left.\frac{\partial^k C}{\partial X^k}\right|_x \tag{3.27}$$

and recall that G_n denotes a current approximation using n points as defined above (3.25), that is, neglecting higher terms. This gives us, for $n = 2$,

$$D_0^1 = G_2 - \tfrac{1}{2} H D_0^2 - \tfrac{1}{6} H^2 D_0^3 - \ldots \tag{3.28}$$

and for $n = 3$,

$$D_0^1 = G_3 + \tfrac{1}{3} H^2 D_0^3 + \tfrac{1}{4} H^3 D_0^4 + \ldots \tag{3.29}$$

and so on for the higher-n forms. An extended table is seen in Bieniasz's paper [108], but these two will suffice here. In order to improve the two approximations G, clearly we need information on the higher derivatives.

One further new notation is useful here, used by Bieniasz. A given current approximation is denoted as $n(m)$, where n is the number of points used to approximate it, and m is the order with respect to H, the intervals in X. Thus, the formulae used so far make, for example, G_2 a 2(1) form and G_3 a 3(2) form. It will be seen that we can easily obtain, for example, 2(3) and 3(4), etc.

In order to obtain the missing higher derivatives, or approximations to them, we write (3.26) for $X = 0$:

$$\left.\frac{\partial^2 C}{\partial X^2}\right|_0 = D_0^2 = F(0, T, C, \partial C/\partial T) = F_0 \ . \tag{3.30}$$

This can be applied directly to (3.28), neglecting the term in D_0^3 there and replacing the term in D_0^2 as in (3.30), obtaining a new approximation,

$$D_0^1 \approx G_2 - \tfrac{1}{2} H F_0 \tag{3.31}$$

which is 2(2), an improvement on the old form. Bieniasz' treatment results in a general equation that can be written as

$$D_0^1 = G_n + H \sum_{i=0}^{n-1} \phi_i F_i \tag{3.32}$$

or, in words, with the old formulae of the above two sections, all $n(n-1)$, improved by the addition of (up to) an equal number n of F_i values with weighting coefficients ϕ_i. In the above simple 2(2) example, we have $\phi_o = -\frac{1}{2}, \phi_1 = 0$. In the table of ϕ coefficients in Bieniasz [108], it is seen that for all n, there is a set of coefficients that give $n(n)$, and they all have the last one, ϕ_{n-1}, equal to zero. These all fail to make use of the last F_{n-1} value, but do have the advantage of an easy calculation of the ϕ coefficients, and easy implementation.

We can go further with the above two-point case, to get 2(3). For this, expressions for F_i are generated from (3.30) by Taylor expansion. Here we use just one:

$$F_1 = F_0 + H D_0^3 + \frac{H^2}{2} D_0^4 + \ldots \tag{3.33}$$

and cutting this off (for the moment) from the fourth derivative onwards, the third derivative is obtained:

$$D_0^3 \approx \frac{1}{H}(F_1 - F_0) \tag{3.34}$$

which, together with (3.30) can be inserted in (3.29) (neglecting the fourth-order derivative term) and upon rearranging, we get

$$D_0^1 \approx G_2 + H\left(-\tfrac{1}{3} F_0 - \tfrac{1}{6} F_1\right) \tag{3.35}$$

which is 2(3), with coefficients $\phi_0 = -\frac{1}{3}, \ \phi_1 = -\frac{1}{6}$.

We can go further, taking the three-point approximation. As well as (3.33), we write the Taylor expansion for F_2:

$$F_2 = F_0 + 2H D_0^3 + \frac{4H^2}{2} D_0^4 + \ldots \ . \tag{3.36}$$

The simpler formula simply makes use of (3.34) and inserting it into (3.36), D_0^4 is obtained and after some rearrangement of the F terms, we get the 3(3) form

$$D_0^1 \approx G_3 + H\left(-\tfrac{1}{3}F_0 + \tfrac{1}{3}F_1\right) \ . \tag{3.37}$$

Here, $\phi_0 = -\frac{1}{3},\ \phi_1 = +\frac{1}{3},\ \phi_2 = 0$ and we thus have the 3(3) form. Again, the last (third) point in F is unused. To involve it as well, write out the Taylor expansions for both F_1 and F_2 to the fourth derivative term:

$$\begin{aligned} F_1 &= F_0 + HD_0^3 + \frac{H^2}{2}D_0^4 + \ldots \\ F_2 &= F_0 + 2HD_0^3 + \frac{4H^2}{2}D_0^4 + \ldots \end{aligned} \tag{3.38}$$

and solve this little system for both derivative terms. One way to do this is to express D_0^3 from the first equation in terms of the other terms and to substitute that in the second. This is a recursive process as described by Bieniasz [108], but what one does is in fact to solve such systems. Doing this, one obtains for this case the 3(4) form

$$D_0^1 \approx G_3 + H\left(-\tfrac{1}{4}F_0 + \tfrac{1}{6}F_1 + \tfrac{1}{12}F_2\right) \ , \tag{3.39}$$

with the coefficients obvious from the formula. This treatment can be extended to higher numbers of points and the reader is invited to look up Table 3 in Bieniasz [108], where this has been done up to $n = 5$, up to the 5(6) form. The results of using these are given in that paper for a number of different electrochemical problems and, not unexpectedly, it seems that the $n(n)$ forms are inferior to the $n(n+1)$ forms, so the latter seem to be the logical choice. There are arguments for the 2(3) form in particular. It is third-order, and this goes well together with higher-order methods, which however rarely recommend themselves, in terms of computing time and programming effort, above that order. Also, with unequal intervals, there are no ready-calculated coefficients for more than two points and thus two points recommend themselves. The 2(3) formula can be applied as it stands in (3.35).

There remains one problem, that of the values of F_i needed for the approximations. Their determination depends on the simulation method used, but at this point, it can be said that the major term, $\partial C/\partial T$, always present, can be approximated simply as

$$\frac{\partial C_i}{\partial T} \approx \frac{C_i - {}'C_i}{\delta T} \tag{3.40}$$

where C_i is the present value, just calculated, and ${}'C_i$ is the last value before the step taken. This is a backward difference, and something better than this can be achieved and is described in Chap. 8. If other terms are contained in F, their most recent values are simply used.

3.7 Second Derivative Approximations

Clearly, some approximations to second derivatives are also needed, although not as many forms. The most widely used approximation is derived as follows. Regard Fig. 3.2 and focus on the point at x_2, where the derivative is to apply. We already have Taylor expansions for the points at x_1 and x_3 (3.7) and (3.10), both of which have a neglected term of $O(h^4)$. Adding the two equations and rearranging leads to the approximation formula

$$y_2'' = \frac{y_1 - 2y_2 + y_3}{h^2} + O(h^2) \,, \tag{3.41}$$

a second-order approximation. Until recently, this has always been used in digital simulation. In 1990, Kimble & White [338] suggested a higher-order formula using five points (and six at the electrode), together with an unusual way of simulating, described in a later chapter. While the method itself is somewhat demanding in terms of computer memory and has not become popular (it solves the whole grid in (X, T) as one large system), the five-point approximation for the second derivative does seem promising, and has been explored [152, 531]. Therefore, both the central five-point scheme and a few asymmetrical multipoint schemes are needed. The procedure is the same as described above for the first derivative. For example, in the case of the central five-point scheme, centered on the point (x_3, y_3) in Fig. 3.2, we write Taylor expansions for the surrounding four points, going out to terms in $h^4 y_3''''$, and solving the system of four equations. In this case, it is the second row of the matrix inverse that provides the coefficients, since we seek y_3'' of the unknowns vector. The result is the matrix equation

$$\begin{bmatrix} -2 & \frac{4}{2!} & \frac{-8}{3!} & \frac{16}{4!} \\ -1 & \frac{1}{2!} & \frac{-1}{3!} & \frac{1}{4!} \\ 1 & \frac{1}{2!} & \frac{1}{3!} & \frac{1}{4!} \\ 2 & \frac{4}{2!} & \frac{8}{3!} & \frac{16}{4!} \end{bmatrix} \begin{bmatrix} h & 0 & 0 & 0 \\ 0 & h^2 & 0 & 0 \\ 0 & 0 & h^3 & 0 \\ 0 & 0 & 0 & h^4 \end{bmatrix} \begin{bmatrix} y_3' \\ y_3'' \\ y_3''' \\ y_3'''' \end{bmatrix} = \begin{bmatrix} y_1 - y_3 \\ y_2 - y_3 \\ y_4 - y_3 \\ y_5 - y_3 \end{bmatrix} . \tag{3.42}$$

Inverting the matrices and multiplying out the second row with the coefficient vector finally yields the approximation, presented in Table A.2 in Appendix A, together with a few others. It turns out that in the process, the terms in h^5 drop out and the final approximation is of $O(h^4)$, arising from the neglected terms in h^6. The formula has been given as early as 1935 by Collatz [169], who also presented some asymmetric forms in his 1960 book [170], and Bickley in 1941 [88]. Noye [423] also provides a number of multipoint second derivatives for use in the solution of *pde*s.

For reasons that become clear in Chap. 9, we also need an asymmetric form, centered on the point (x_2, y_2), since in a simulation (index 1 being the electrode) this point is also subject to diffusional changes. The obvious course here is to use an asymmetric five-point formula, but this, as pointed

out by Collatz [170], is only of $O(h^3)$. Presumably for this reason, Kimble & White chose a six-point asymmetric scheme here, not provided by Collatz (who goes to a seven-point scheme). The six-point scheme is indeed $O(h^4)$ and is included in Table A.2. At the outer end of the diffusion space, this is needed again at the second-last point and it is given as $y_5''(6)$ in the Table, meaning the second derivative applied to point (x_5, y_5). The coefficients are those for $y_2''(6)$ in reverse order but without sign flip.

3.8 Derivatives on Unevenly Spaced Points

Some simulation techniques make use of points along x, and indeed sometimes along t, that are spaced unevenly, either in some smooth transformational progression or more or less arbitrarily. A general treatment is given in this section, as well as a few particular algebraic solutions.

The need for such formulae and algorithms became clear upon publication of the paper by Rudolph [478], showing that direct discretisation of derivatives on an exponentially expanding grid is in fact better than discretisation on an equally spaced transformed grid. This is in contrast with what the computer science community is agreed upon, based on early works [186,328,422]. One reason for this is, as Rudolph points out, that the concentration profiles in electrochemical work are normally almost linear near the electrode, so that current approximations or other first derivative expressions in that region in fact operate on a curved function in transformed space. This does not explain why second derivatives in the diffusion space, too, are more accurate when discretised directly on an unequal grid, as they have been found to be by some numerical experiments. There have been a few publications lately presenting derivative approximations on unequal grids in the form of algebraic solutions [149, 260, 478]. Bieniasz, in his introductory paper [112] on adaptive grids, used a 4-point formula for a second derivative, which was that of Blom [122]. This was found later to be inconsistent, and Bieniasz presented a corrected expression [96]. Britz and Strutwolf [153] showed a derivation of such formulae and a few particular examples. For approximations on just a few (3 or 4) points, such formulae can be useful but for higher-order forms, a numerical approach is better.

Figure 3.3 shows a few points along some function $u = f(x)$. The symbol u is used for this, to indicate that intervals are unequal. The points are again numbered from 1 to n in general, and a given derivative might be referred to a point at index i among the n points. They will, as above, following the convention for equal intervals, be denoted by the symbols $u_i'(n)$ for first derivatives or $u_i''(n)$ for second derivatives. A one-sided first derivative, for example a current approximation, will then be $u_1'(n)$ and a central second derivative as is often employed, referring to the middle point, is $u_2''(3)$. First derivatives will be given in terms of linear sums of the form

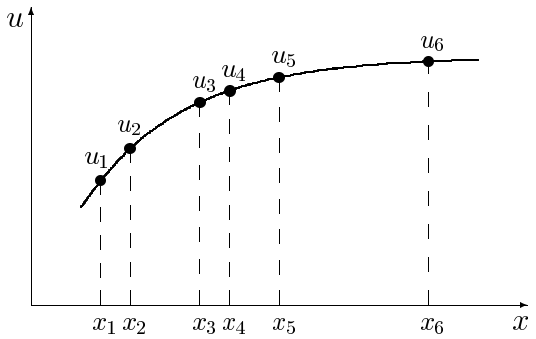


Fig. 3.3. Arbitrary function with unevenly spaced points

$$u' = \sum_{i=1}^{n} \beta_i \, u_i \tag{3.43}$$

and second derivatives by a similar expression,

$$u'' = \sum_{i=1}^{n} \alpha_i \, u_i \; . \tag{3.44}$$

The coefficients are used to compute the derivatives, but can also be useful in the discretisation of derivative boundary conditions or in the setting up of discretisation matrices in some problems.

Define a sequence of displacements $h_k, k = 1 \ldots n$, given by

$$h_k = x_k - x_i \tag{3.45}$$

where the reference point x_i is fixed. Clearly, $h_i = 0$. This is the set of displacements from the reference point. Taylor expansions are written for all points around the ith point, which is the reference point. Derivatives higher than the second are involved in the expansion, and $D^j u$ denotes the jth derivative operator on u. For the kth point, we have

$$u_k = u_i + h_k D u_i + \frac{h_k^2}{2!} D^2 u_i + \frac{h_k^3}{3!} D^3 u_i + \ldots, + \frac{h_k^{n-1}}{(n-1)!} D^{n-1} u_i + O(h_k^n) \; . \tag{3.46}$$

Just $n - 1$ derivatives are needed on the right-hand side, and the dominant error term is indicated. The resulting system can be cast in vector/matrix form

$$\boldsymbol{H}\boldsymbol{d} = \boldsymbol{r} + \boldsymbol{e} \tag{3.47}$$

where

$$\boldsymbol{H} \equiv \begin{bmatrix} h_1 & h_1^2 & h_1^3 & \dots & h_1^{n-1} \\ h_2 & h_2^2 & h_2^3 & \dots & h_2^{n-1} \\ & & \dots & & \\ h_{i-1} & h_{i-1}^2 & h_{i-1}^3 & \dots & h_{i-1}^{n-1} \\ h_{i+1} & h_{i+1}^2 & h_{i+1}^3 & \dots & h_{i+1}^{n-1} \\ & & \dots & & \\ h_n & h_n^2 & h_n^3 & \dots & h_n^{n-1} \end{bmatrix} , \tag{3.48}$$

$\boldsymbol{d}$ denotes the vector of derivatives $[Du_i \;\; \frac{D^2 u_i}{2!} \;\; \dots \;\; \frac{D^{n-1} u_i}{(n-1)!}]^T$, $\boldsymbol{r}$ stands for the vector of the knowns $(u_k - u_i)$, and $\boldsymbol{e}$ for the vector of the last, error terms in (3.46). The vector $\boldsymbol{d}$ has been chosen as it was, with the factorials glued to the terms, because this avoids the very small inverse factorials in the matrix. The matrix $\boldsymbol{H}$, as it is, already has elements of greatly varying magnitudes because of the powers of intervals, which can be small; this leads to inaccurate inversion, and the problem might be compounded by the factorials. These must be multiplied by appropriately after inversion.

Matrix $\boldsymbol{H}$ can be automatically generated and its inverse then yields the solution for the derivatives. We require only the first two derivatives, which arise from the first two rows of the inverse. If the inverse be $\boldsymbol{V}$ and its elements at indices i, j be $v_{i,j}$, then we have, by expanding $\boldsymbol{d} = \boldsymbol{V}\boldsymbol{r}$ (neglecting the error terms $\boldsymbol{V}\boldsymbol{e}$ for the moment),

$$u_i'(n) = Du_i = v_{1,1}(u_1 - u_i) + v_{1,2}(u_2 - u_i) + \dots + v_{1,n-1}(u_n - u_i) \; . \tag{3.49}$$

There is no term in $v_{1,\,i}(u_i - u_i)$ and so there is a break at index i in the sequence of terms. For $k < i$, the terms are $v_{1,\,k}(u_k - u_i)$, while for $k > i$, the terms are $v_{1,k-1}(u_k - u_i)$. Comparing with (3.43), it is clear that the row of $v_{1,\,k}$ represents the β_k coefficients, bar β_i. We have

$$\beta_k = \begin{cases} v_{1,k} \; , & k < i \\ v_{1,k-1} \; , & k > i \end{cases} \tag{3.50}$$

(omitting β_i). Finally,

$$\beta_i = - \sum_{\substack{k=1 \\ (k \neq i)}}^{n} \beta_k \; . \tag{3.51}$$

Similarly, the α coefficients in (3.44) are obtained from the second row of the inverse, so that

$$\alpha_k = \begin{cases} 2!\; v_{2,k} \; , & k < i \\ 2!\; v_{2,k-1} \; , & k > i \end{cases} \tag{3.52}$$

(remembering to multiply by 2!) and

$$\alpha_i = -\sum_{\substack{k=1 \\ (k\neq i)}}^{n} \alpha_k \ . \tag{3.53}$$

The above has been programmed as a general subroutine `U_DERIV` (see Appendix C), which returns both the wanted derivative (first or second) as well as the coefficients that produced it.

3.8.1 Error Orders

From the above treatment, the error orders of the approximations can be determined. First, a definition of what is meant here is required. With equal intervals of length h, orders are expressed as powers of that length. Here we have arbitrarily spaced points, and thus a set of different h_k. In computations to confirm error order expectations, the following scheme can serve. Refer all h_k as displacements from point i, as above (3.45). A given derivative can then be computed. Then, all points around the reference point x_i are moved to a given fraction a of their original displacements from the reference point, so that now there is a new set of displacements,

$$h'_k = ah_k \ , \tag{3.54}$$

new values are set at the new set of points and a new derivative is computed. The two derivative estimates then yield the order, as usual.

Another way of expressing this is to take the average of all h_k, calling this simply h. The Taylor expansions (3.46) then each contain an error term of $O(h^n)$, which becomes the vector $\boldsymbol{e}$ in (3.47). When producing the respective derivative by multiplying the first or second row of the inverse matrix with the vector of knowns, an error term will arise by the multiplication of the same row with vector $\boldsymbol{e}$. Some consideration of matrix $\boldsymbol{H}$ and its inverse $\boldsymbol{V}$ reveals that the top row of $\boldsymbol{V}$ (which gives the first derivative) consists of elements, all of which are $O(h^{-1})$ and this, multiplied with the error terms, results in an error of $O(h^{n-1})$. Similarly, the second row, which gives the second derivative, has all terms of $O(h^{-2})$, so that second derivatives are $O(h^{n-2})$ accurate.

Some numerical tests show that for first derivatives using n points, the error is indeed of order $n-1$, while for second derivatives, it is of order $n-2$. If the intervals are equal and the approximations are central (this is possible only for odd n), the order goes up by unity for both derivatives.

Theoretically, there is no limit on the number n of points used in the approximations. In practice, however, a limit is set by roundoff in the computations, making an increase in n useless, and the factorials in the matrix $\boldsymbol{H}$ will increase to impractical levels. In any case, there seems little point in n values greater than about 8, although for the usual 32-bit computers in use today, up to 12-point formulas can be accommodated.

3.8.2 A Special Case

The above has considered arbitrarily spaced grids, whereas in practice, the spacing is often that of Feldberg [231]. In terms of points, the special case is a sequence of positions given by an exponentially expanding series of spatial intervals. This will be detailed in Chap. 7. Here, it is sufficient to mention that this special case makes the derivation of the coefficients for various derivative approximations easier and the expressions themselves more compact, as was reported by Martínez-Ortiz [385]. That author also found that there is a particular value for the expansion parameter, $\gamma = \sqrt{2}$, for which the asymmetric four-point second derivative, referred to the second of the four points $u''(2, 4)$, is third-order accurate, rather than second-order as for other parameter values or arbitrary placement of points. This can be of use in simulation. The four-point approximation has some good properties besides this, as will be explained in Chap. 7.

3.8.3 Current Approximation

The above treatment includes the current approximation on an unequal grid, and the subroutine `U_DERIV` can compute it. It is, however, a little unwieldy, and a simpler interface to it is also mentioned in the same Appendix, function `GU`, which only requires the three arguments `(C, x, n)`. Similarly, the function `CU` computes C_0 from a given concentration profile and a known current G.

It should be noted here that Bieniasz [93] used what amounts to our $\mathcal{G}$, designed for equal intervals, to calculate current approximations on an arbitrary grid. The idea is that the spatial axis is mapped onto an imagined, equally spaced, new axis and the approximation then becomes, using the notation of Sect. 3.5,

$$G = \left(\sum_{i=0}^{n-1} \beta_i u_i \right) \left(\sum_{i=0}^{n-1} \beta_i x_i \right)^{-1} = \frac{\mathcal{G}(\boldsymbol{u}, n, 1)}{\mathcal{G}(\boldsymbol{x}, n, 1)} \,. \tag{3.55}$$

Interesting though this trick is, the results using it are disappointing.

3.8.4 A Specific Approximation

For a small number n of points, it may be worthwhile using the algebraic solutions for the coefficients. The procedure is as described above, but instead of inserting actual h_k values into the matrix in (3.48), that matrix is inverted algebraically and the coefficients expressed as a general formula. These are given, for a few approximations, both for first and second derivatives (restricted to those that are deemed to be of practical interest) in Appendix A. All the current approximations up to $n = 4$ are provided there, as well as

some second derivatives up to the same limit. The author has derived forms up to $n = 6$, and it soon becomes clear that the disadvantage of these is that every specific $u_i'(n)$ or $u_i''(n)$ requires its own expression set, and for the larger n, the resulting subroutines become rather long.

To give an idea of how the tabulated formulae are derived, the derivation for the "central" second derivative on three points, $u_2''(3)$ is shown here. We have the two h displacements h_1 and h_3 (see their definitions in (3.45)), and the matrix $\boldsymbol{H}$ to be inverted is then

$$\boldsymbol{H} = \begin{bmatrix} h_1 & \frac{h_1^2}{2!} \\ h_3 & \frac{h_3^2}{2!} \end{bmatrix} . \tag{3.56}$$

This is easily manually inverted. The determinant is

$$\Delta = \tfrac{1}{2}\left(h_1 h_3^2 - h_3 h_1^2\right) \tag{3.57}$$

and the inverse $\boldsymbol{V}$ is then

$$\boldsymbol{V} = \frac{1}{\Delta}\begin{bmatrix} \frac{1}{2}h_3 & -\frac{1}{2}h_1 \\ -h_3 & h_1 \end{bmatrix} . \tag{3.58}$$

The solution is then the same as (3.52) and (3.53), which directly leads to the coefficients for the linear sum,

$$u_2''(3) = \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 \tag{3.59}$$

with

$$\alpha_1 = \frac{-2}{h_1(h_3 - h_1)}$$

$$\alpha_2 = -(\alpha_1 + \alpha_3) = \frac{2}{h_1 h_3} \tag{3.60}$$

$$\alpha_3 = \frac{2}{h_3(h_3 - h_1)} . \tag{3.61}$$

In all these cases, the coefficient for the reference point is the negative sum of all the others, as is clear from the form of (3.52) and (3.53). A similar formula has been given by Gavaghan [260], except that his notation for the displacements was such that all distances from the reference point are positive, so that the final formula differs in the signs of some of the terms.

4 Ordinary Differential Equations

In this chapter, the numerical solution of ordinary differential equations (*ode*s) will be described. There is a direct connection between this area and that of partial differential equations (*pde*s), as noted in, for example [558]. The *ode* field is large; but here we restrict ourselves to those techniques that appear again in the *pde* field. Readers wishing greater depth than is presented here can find it in the great number of texts on the subject, such as the classics by Lapidus & Seinfeld [351], Gear [264] or Jain [314]; there is a very clear chapter in Gerald [266].

We begin with single *ode*s. At the end of this chapter, systems of *ode*s are dealt with, as they are in fact one way of handling *pde*s, using the Method of Lines (MOL, see Chap. 9), which has a system of *ode*s as an intermediate stage, or something close to it.

The kind of *ode*s most relevant in the present context is of the form

$$y' = f(y) \tag{4.1}$$

with the boundary condition

$$y(t = 0) = y_0 \, . \tag{4.2}$$

There is a more general form in which the time variable t also appears in the brackets on the right-hand side of (4.1), but in the present context, it almost never does. The simplified form will be our model.

In what follows below, the discussion assumes that a point $y(t_n)$ at time $t_n = n\delta t$ is known (as well as previous points), and that we wish to calculate the next point $y(t + \delta t)$ or $y(t_{n+1})$. These will also be denoted by y_n, y_{n+1} etc, interchangeably with the other notations.

4.1 An Example *ode*

In what follows, the following specific *ode* will often be used as an example:

$$y' = -y \tag{4.3}$$

with the single boundary condition

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 51–71 (2005)
www.springerlink.com

$$y(0) = 1\ . \tag{4.4}$$

In numerical texts, the right-hand side of (4.3) often has a multiplier, but this can be normalised out. We note that this is an instance of (4.1) with $f(y) = -y$. It has a known solution,

$$y(t) = \exp(-t) \tag{4.5}$$

and is a very convenient *ode* with which to test methods. Here it will be used to illustrate the implementations.

4.2 Local and Global Errors

A note is in order here on errors in the numerical solution of an *ode*. There are (regarding errors in a certain light) two kinds of errors. One is the **local error**, being the error added by a single step. The solution is always carried forward to a final point in t, using a number N of steps, and at that point we have a final, or **global error**. Unfortunately, this is always of a lower order than the local error.

4.3 What Distinguishes the Methods

For most of the methods used to solve *ode*s (at least, those described here), the way in which the methods differ hinges on how the following three questions are answered:

1. How is y' approximated (how many points are used)?
2. To what value of t is that approximation intended to apply?
3. How is the right-hand side of (4.1) expressed or approximated?

It is very important to be clear on these points in devising simulation strategies, especially (when going on to *pde*s) the boundary conditions, which must conform to these points as well.

It will be seen that for the three methods Euler, BI and the trapezium method, the same approximation expression is used for the left-hand side of (4.1) but because of points made in questions (2) and (3) above, the methods are very different.

4.4 Euler Method

Consider Fig. 4.1. The curve is the underlying function $f(y)$ that we are trying to find and we have two pieces of information: one point on the curve, here at $t = 0$, the fat point marked in the figure, and the gradient at any

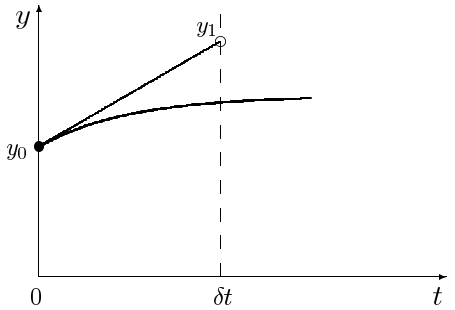


Fig. 4.1. The Euler method

point, for example, at the same point, drawn as a tangent. The procedure is now to find a point y_1, at a subsequent chosen time t, for example δt, as shown. The picture represents the mathematical problem of finding the solution to the *ode*, (4.1).

The simplest way to find other points on the curve, or approximations to such points, is to move along the tanget drawn, to $t = \delta t$, as shown in the figure. Clearly, this will not land on the curve, if it is indeed curved as shown, but hopefully somewhere close to it, producing the new point y_1. This will then be repeated, because from (4.1) we can obtain a new tangent, using y_1 and so on. If we have chosen δt not too large (much smaller, for example, than in Fig. 4.1), this will result in a series of discrete points $y_i, i = 1, 2, \ldots, N$ that will be an approximation to the desired solution. The method just described is the **Euler method** and is the basis for what in digital simulation is called the **explicit method**.

Expressing this mathematically, y' is approximated by the simple two-point formula (3.1), written as

$$y'(t) = \frac{y(t + \delta t) - y(t)}{\delta t} + O(\delta t) \; . \tag{4.6}$$

It turns out that the order of this approximation is also the global error order of the calculation using the Euler method. An alternative way to proceed is to go from the Taylor expansion for $y(t + \delta t)$, as in (3.3),

$$y_{n+1} = y_n + \delta t \, y'_n + O(\delta t^2) \tag{4.7}$$

and substituting for y' from (4.1),

$$y_{n+1} = y_n + \delta t \, f(y_n) + O(\delta t^2) \tag{4.8}$$

which we note is $O(\delta t^2)$. The order refers to the local error due to a single step. In [266] there is clear derivation of the order of the global error from that of the local error, which is not reproduced here. Broadly, the idea is that when taking N steps of length δt, each contributing a local error of $O(\delta t^2)$,

the order is reduced to $O(\delta t)$, the same as the order of the approximation of the derivative, as seen in (4.6).

What we have here, with the Euler method, is the definition of the derivative as pertaining to time t (or $n\delta t$) and thus $f(y(t))$ or $f(y_n)$ on the right-hand side. For our specific example (4.3), this becomes approximately

$$y_{n+1} = y_n - \delta t\, y_n \tag{4.9}$$

or

$$y_{n+1} = y_n(1 - \delta t)\ . \tag{4.10}$$

4.5 Runge-Kutta, RK

Figure 4.2 illustrates an improvement on the Euler method. The point marked by $\times$ is the same as point y_1 in Fig. 4.1, having moved up the tangent from $t = 0$, line 1 in the present figure. This point is just an intermediate result. Using (4.1), we calculate the slope y', that is, $f(y_1)$ and draw that slope, line 2. We can hope that it will be an approximation to the slope at δt. We now have two slopes, lines 1 and 2. Drawing a third slope midway between these two, (dashed) line 3, might be a better approximation to the slope we should have used, and we use it now, line 4, parallel to line 3, starting from the point y_0, and this line hits the δt line at the new point y_1. This turns out to be a much better approximation to the underlying curve. The reason for this is that the slope of line 4 closely approximates the slope which, if we had known it, would have given us the exact solution, namely that of the line drawn from y_0 to the point on the underlying curve at δt. The above describes the second-order **Runge-Kutta** (**RK**) method (there are other, higher-order variants of Runge-Kutta). This is still an explicit method; the word "explicit" means that at each step, the new point is calculated from previously calculated points only.

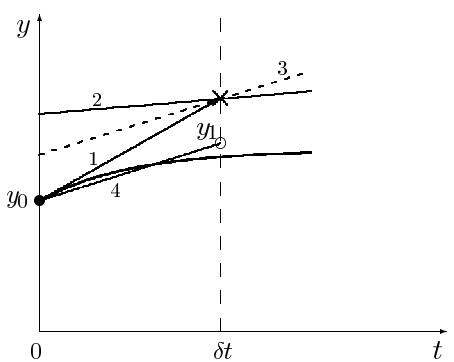


Fig. 4.2. Second-order Runge-Kutta

For the mathematics of this, consider the discrete equation resulting from the Euler method, as in (4.8). Note that the new point, y_{n+1} is formed from the old point y_n by the addition of a term, here $\delta t f(y_n)$. With RK, these terms are given the symbols k_i, there are from one to several of them, and they are added in a weighted manner. The procedure is to generate a number of these k's. One begins with an Euler step,

$$k_1 = \delta t f(y_n) \; . \tag{4.11}$$

(Note that the Euler method can be regarded as a first-order RK form, if we write (4.8) as

$$y_{n+1} = y_n + k_1 \tag{4.12}$$

which is the same thing). Following the description of Fig. 4.2, the mathematical procedure for second-order RK is now to generate k_2 from the tentative point at $y_n + k_1$:

$$k_2 = \delta t f(y_n + k_1) \tag{4.13}$$

and then the final corrected point is

$$y_{n+1} = y_n + \tfrac{1}{2}(k_1 + k_2) \; . \tag{4.14}$$

This formula has a global error of $O(\delta t^2)$ and will here be called **RK2**. We can do even better, generating more k's and getting higher orders. All these RK formulae, including RK2, have variants that have the same error orders. For example, RK2 can also be carried out by generating k_2 as

$$k_2 = \delta t f\left(y_n + \tfrac{1}{2}k_1\right) \tag{4.15}$$

following with

$$y_{n+1} = y_n + k_2 \; . \tag{4.16}$$

Here, only certain of all the variants are given. One variant of third-order RK uses k_1 as defined above (4.11), then (4.15) for k_2, and finally a third,

$$k_3 = \delta t f(y_n - k_1 + 2k_2) \tag{4.17}$$

giving the third-order scheme **RK3**

$$y_{n+1} = y_n + \tfrac{1}{6}(k_1 + 4k_2 + k_3) \; . \tag{4.18}$$

Numerical professionals, when using the term "Runge-Kutta", usually mean fourth-order RK, and the classical scheme, here **RK4**, is k_1 as above, then k_2 as in (4.15), then

$$k_3 = \delta t f(y_n + \tfrac{1}{2}k_2) \; , \tag{4.19}$$

then a fourth k,

$$k_4 = \delta t f(y_n + k_3) \tag{4.20}$$

and finally the result, of global fourth-order

$$y_{n+1} = y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \; . \tag{4.21}$$

These formulas can all be applied to *pdes* in a simple manner, easy to program, but have certain drawbacks, as described in a later chapter.

4.6 Backwards Implicit, BI

Another possibility is to let the same derivative approximation pertain to the next time; this is the **backward implicit** (**BI**) method:

$$y_{n+1} = y_n + \delta t f(y_{n+1}) \tag{4.22}$$

which again leads to a global error order $O(\delta t)$, and becomes, for our specific example (4.3), rearranging,

$$y_{n+1} = y_n \frac{1}{1+\delta t} \ . \tag{4.23}$$

This method seems at first sight unpromising, because of its low error order, the same as that for Euler. However, it has some very useful stability properties (see later) and forms the basis for several high-order methods, as will be seen.

4.7 Trapezium or Midpoint Method

We know from (3.13) in Chap. 3, how that same derivative approximation is of higher order $O(\delta t^2)$ when applied at the midpoint, and this leads to the **trapezium** method or **midpoint rule**, in which we must find an expression for the right-hand side of (4.1) at time $t + \frac{1}{2}\delta t$. This can be approximated as the average of the values at both ends:

$$f(y(t + \tfrac{1}{2}\delta t)) \approx \frac{f(y(t)) + f(y(t+\delta t))}{2} \tag{4.24}$$

giving

$$y_{n+1} = y_n + \delta t \frac{f(y_n) + f(y_{n+1})}{2} \ . \tag{4.25}$$

This can be awkward to go on with, being implicit in y_{n+1}; in our specific example (4.3), however, there is no problem, the above equation becoming

$$y_{n+1} = y_n - \delta t \, \frac{y_n + y_{n+1}}{2} \tag{4.26}$$

or, after rearranging,

$$y_{n+1} = y_n \frac{1 - \frac{1}{2}\delta t}{1 + \frac{1}{2}\delta t} \tag{4.27}$$

which turns out to be (global) $O(\delta t^2)$. It is the basis of the Crank-Nicolson method when applied to *pde*s, as will be seen.

4.8 Backward Differentiation Formula, BDF

The **BDF** method is ascribed to Curtiss & Hirschfelder [188], who described it in 1952, although Bickley [88] had essentially, albeit briefly, mentioned it already in 1941. Considering Fig. 4.3, the method can be seen as a multipoint extension of BI; the derivative y' is formed by using a number k of points from y_{n-k+2} to y_{n+1}, but referred to the new point y_{n+1}. This implies a backward derivative, with formulas of the form $y'_n(n)$ as in Appendix A, Table A.1. For example, using the three points shown in Fig. 4.3 (in other words, $k = 3$), the table yields the formula

$$y'_{n+1} \approx \frac{\frac{1}{2}y_{n-1} - 2y_n + \frac{3}{2}y_{n+1}}{\delta t} \,. \tag{4.28}$$

For this value of k, then, (4.1) is discretised as

$$y_{n-1} - 4y_n + 3y_{n+1} = 2\delta t f(y_{n+1}) \tag{4.29}$$

(note that the case $k = 2$ is equivalent to BI). This is implicit, as with BI. For our example (4.3), the function on the right-hand side is simply $-y_{n+1}$ and rearrangement then produces

$$y_{n+1} = \frac{-y_{n-1} + 4y_n}{3 + 2\delta t} \tag{4.30}$$

which is of (global) $O(\delta t^2)$. The order can be increased by increasing k, the number of time levels (points) used for the backward difference approximation (see Table A.1 for the coefficients $y'_n(n)$).

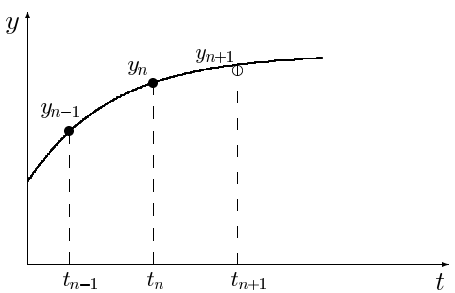


Fig. 4.3. BDF

It turns out that although the BDF schemes achieve higher and higher orders as k increases, the solution begins to oscillate (certainly when the method is adapted to *pde*s) at about $k = 5$ and becomes unstable for $k > 7$. As applied to diffusion simulations by the Feldberg school [236, 402], a value of 5 is normally used. The choice of this parameter is discussed in a later chapter.

Note that the parameter k as defined here, being the number of time points used for the backward difference, which is the convention in electrochemistry since [402], differs from the usage in computer science, where k refers to the number of intervals ("levels") between these points, and is thus smaller by one. It is the electrochemical usage that is adhered to in this book.

4.8.1 Starting BDF

BDF presents the problem of how to start it. If using, for example, a 5-point formula, it is not possible to use it for time points earlier than $4\delta t$. At earlier times, an insufficient number of points for the application of the formula is known. This problem is mentioned in, among others, [129, 284, 302]. There are various ways of dealing with this, and four possible strategies will be described here. The simplest way is to ignore the problem, and to artificially assume some points at times $t < 0$, all equal to the initial value given for $t = 0$, and starting directly, generating the point at δt. This is the **simple start**, which is favoured by Feldberg and coworkers [402], originally without justification other than convenience (private communication, Feldberg 2001). It is illustrated graphically in Fig. 4.4. In the figure, the vertical height of the points indicates the time level, the base line being the level $t = 0$. Filled points are known values while empty points are those to be calculated. Four steps are shown for a sequence using $k = 4$. The sequence starts with the set to the far left, where the three known points are those for $t = 0$. The next set uses two of these and the one just calculated. After this, all required points for subsequent iterations are available. Surprisingly, it turns out, that although this yields rather poor results in itself (as will be seen below), a small trick used by Feldberg turns it into a highly accurate method. The trick consists in correcting the time value given to each completed iteration by subtracting from it half a time interval. That is, the iteration numbers $1, 2, \ldots$ which normally are taken as indicating the times $\delta t, 2\delta t, \ldots$, are taken to indicate the values $\frac{1}{2}\delta t, \frac{3}{2}\delta t, \ldots$. This will be called the **simple start with correction**, to be described later.

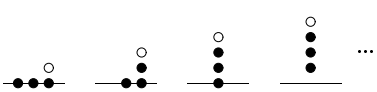


Fig. 4.4. Simple BDF start schematic

Among computing professionals solving *ode*s, the usual practice has been what might be called the **rational start**, see Fig. 4.5. This starts with the method BI, which can be regarded as 2-point BDF, to generate the first point, then uses 3-point BDF to generate the next, then 4-point, and so on, until the desired k has been reached, and continues from there. Inevitably, the first few points will then have errors of a lower order than later points.

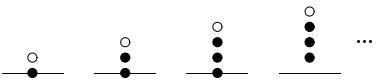


Fig. 4.5. Rational BDF start schematics

This does a little better in terms of accuracy than the simple start (without the correction).

There is a high-order start provided by the method of Kimble & White, which will be called the **KW** method here. The method was originally designed [338] for the solution of partial differential equations, computing a whole grid in space and time in one large system of equations. It is described as applied to *ode*s below, in detail, in Sect. 4.10. In the context of starting a BDF iteration, it can be applied to solve for the first $k-1$ new values as a system of equations. It is illustrated in Fig. 4.6. The figure shows that (again, $k=4$) all three unknown points are calculated at once. Briefly, what is done is to apply, at each of the unknown time levels, a four-point approximation to the derivative referred to that level; in the present case (Fig. 4.6), all three are asymmetrical forms, being the three $y_i'(4), i=2,3,4$ of Table A.1 in Appendix A. This gives $k-1$ (here, 3) equations in as many unknowns. This is a truncated application of the general Kimble & White method described below.

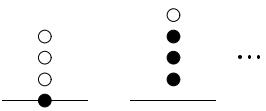


Fig. 4.6. KW BDF start schematics

Time Shifts

We allow ourselves a short digression here, in order to make a special point. There are two ways of presenting an error in a numerical solution of a differential equation. The usual way is to refer to the error in the quantity computed at each new time interval; that is, the difference between the numerical approximation and the exact solution (if it is known). Another way is to compute, for each calculated value, the time at which that value is exact, and to express the error as a time shift, the difference between the calculated time and the exact time at that iteration number. It is called a time shift because in many kinds of simulations dealt with in this book, time itself does not enter the equations and, once a simulated sequence of values has become shifted along in time, that shift is permanent. Putting this another way, there is no clock inherent in the method. It will be seen (Chap. 8) that in fact, in

cyclic voltammetry, it does enter the equations indirectly, and no time shifts are encountered in these simulations.

It turns out that there are characteristic time shifts associated with the various methods for solving differential equations [140]. For example, both the Euler method and backwards implicit have a linearly increasing time shift, reaching respectively $+\frac{1}{2}\delta t$ and $-\frac{1}{2}\delta t$ at (normalised) $t = 1$, whereas a higher-order method such as the trapezium method has a time shift close to zero. It was found [140] that for BDF with the simple start, the time shift appeared to converge to $-\frac{1}{2}\delta t$. This seemed to justify empirically the Feldberg time correction, but there was no explanation for the effect at the time. Such an explanation was found recently [142, 155] and the proof is reproduced in Appendix B. Remarkably, for any function on the right-hand side of a differential equation such as (4.1), whether ordinary or partial (as long as time is not a parameter in the function), and for any BDF value of k, there is a convergence to a time shift of exactly $-\frac{1}{2}\delta t$. It will be seen that this makes the Feldberg starting protocol for BDF a very efficient way of applying BDF.

Testing the Starting Protocols

To illustrate the points made above, some test computations were run, solving the usual *ode* (4.3). Ten steps were taken over the interval $0 < t \leq 1$, using BDF with $k = 4$. Fig. 4.7 shows the results, plotting the error against iteration number. The simple start produces the largest error (curve 1), followed by the rational start (curve 2). The KW start is a line that might be mistaken for the zero axis (curve 3). Remarkably, curve 4, showing the simple start with the time correction, converges to the second-best error of the four. Since this is also as easy to program as the simple method, it is clearly preferable. It was found by some numerical experiments that this method is also the most efficient for electrochemical digital simulations [154]. In that work, the KW method was optimised, and indeed produced highly accurate results. However, this was at considerable computational cost. Efficiency is determined in terms of achieving a target accuracy using the shortest possible computing time, and in these terms, the simple start with the correction was the most efficient. This should be received with some relief, as the KW method is not trivial to implement with *pde*s, especially when optimised for speed, using sparse matrix techniques.

Finally, Lambert [344] describes a high-order start for general multi-level methods, based on Taylor expansions using higher derivatives. This seems less practical to use as, for example, KW.

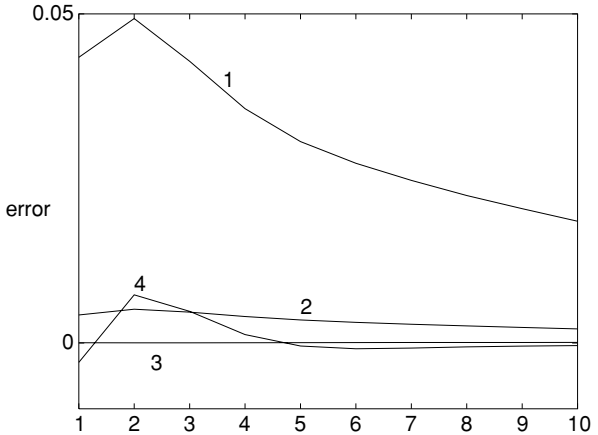


Fig. 4.7. Errors for the four starting methods (see text)

4.9 Extrapolation

Extrapolation is an old technique invented by Richardson in 1927 [469]. Generally it makes use of known error orders to increase accuracy. In the present context, its application is based on the first-order method BI, mentioned above. One defines a notation in terms of operations L on the variable $y(t)$, the operation being that of taking a step forward in time. Thus, the notation $L_1 y(t)$ or, in terms of discrete time steps where one whole interval is δt, $L_1 y_n$, means a single step of one interval (the 1 being indicated by the subscript on L). The simplest variant is then the application of operation L_1, followed by two operations, $L^2_{1/2}$, that is, two consecutive steps of half δt (again starting the first from y_n), and finally a linear combination of the two results:

$$y_{n+1} = \left(2L^2_{1/2} - L_1\right) y_n \ . \tag{4.31}$$

The reason why this provides a better estimate of y_{n+1} is that the (global) error e from a series of single steps of size δt is a polynomial

$$e = a_1 \delta t + a_2 \delta t^2 + \ldots \tag{4.32}$$

with the shown (unknown) coefficients. Clearly, (4.31) will eliminate the first term in that polynomial, leaving only higher terms. The above scheme thus provides an estimate for y_{n+1} that is $O(\delta t^2)$.

Likewise it is possible to eliminate even higher order error terms as well, by using more complicated combinations of step sequences. A full description of these is given by Lawson & Morris [356] (second-order only) and Gourlay & Morris [277]; these authors adapted the method to the solution of *pde*s and more is said on that in a later chapter. With the higher-order forms, there are again variants, as with Runge-Kutta. Gourlay & Morris carried out some

analyses and numerical experiments and the two "winning" schemes are as follows. The third-order scheme is

$$y_{n+1} = \left(\tfrac{9}{2}L_{1/3}^3 - \tfrac{9}{2}L_{2/3}L_{1/3} + L_1\right) y_n \tag{4.33}$$

where the sequence $L_{2/3}L_{1/3}$ means one step of $\frac{2}{3}\delta t$ followed by one more of $\frac{1}{3}\delta t$. The best fourth order scheme is

$$y_{n+1} = \left(8L_{1/4}^4 + \tfrac{40}{9}L_{3/4}L_{1/4} - \tfrac{32}{3}L_{1/2}L_{1/4}^2 - \tfrac{7}{9}L_1\right) y_n \ . \tag{4.34}$$

In the literature, the schemes are usually described not in terms of fractional steps but with a number of whole-interval steps; the two descriptions are equivalent, however, and it seems that a combination of fractional steps, ending with a new value at the next time interval, is more convenient.

4.10 Kimble & White, KW

The method due to Kimble & White [338] is not actually a method designed for *ode*s, but was devised by the authors for electrochemical *pde*s. The method can however be easily adapted to *ode*s and in fact might be more appropriate there. The method described in 1990 had a precursor in 1987 [414] and this section will start with a description of its expression for *ode*s, because it is simpler and makes the point more clearly. A cut-down application of it has already been outlined in Sect. 4.8.1.

The essence of KW is that multi-point central differences are used as derivatives along most of the t scale, with some asymmetric expressions necessarily added at the ends. Rather than using the time-marching method that is common to all the methods described in previous sections, KW puts all the approximations into one large system of equations, and solves the lot. It turns out that this results in a fortuitous stability [141].

The method is based on another time-marching scheme not mentioned in the above sections: the **leapfrog method**, using central differences. Equation (4.1) can be approximated as

$$\frac{y_{n+1} - y_{n-1}}{2\delta t} = f(y_n) \tag{4.35}$$

where the derivative y' is formed from the central difference spanning two time intervals, and is referred to time t_n. This seems intuitively satisfactory and indeed the resulting formula,

$$y_{n+1} = y_{n-1} + 2\delta t f(y_n) \ , \tag{4.36}$$

has a local error of $O(\delta t^3)$ and a global error of $O(\delta t^2)$. Its "only" drawback is that, used in this manner (time-marching), it is unstable and the solution

soon diverges. In the KW scheme, however, applied at (almost) all points in time, it generates an equation system. Clearly, if we have N points in time for which we want values of y_i, starting with the initial value y_0, we can write expressions like (4.35) for all $y_i, i = 1 \ldots N-1$. There is an expression missing for the point at $N\delta t$, for which there is no central difference. Here, Nguyen & White [414] would use a BDF form and in this case, a three-point form is appropriate, as it has the same error order as the central difference.

The equations then are the following. For $i = 1$ we have, for the general *ode* (4.1),

$$\frac{y_2 - y_0}{2\delta t} = f(y_1) \tag{4.37}$$

which includes the boundary value y_0. For points $y_i, i = 2 \ldots N-1$, we have

$$\frac{y_{i+1} - y_{i-1}}{2\delta t} = f(y_i) \tag{4.38}$$

and for the last point, the BDF form

$$\frac{y_{N-2} - 4y_{N-1} + 3y_N}{2\delta t} = f(y_N) \tag{4.39}$$

completing the system, which can now be arranged suitably, writing out the functions $f(y)$. In the present example *ode* (4.3), we have $f(y_i) = -y_i$, and the system becomes

$$\begin{bmatrix} 2\delta t & 1 & & & \\ -1 & 2\delta t & 1 & & \\ & \ddots & & \ddots & \\ & & -1 & 2\delta t & 1 \\ & & 1 & -4 & (3+2\delta t) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} y_0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} . \tag{4.40}$$

The interesting thing is that all but the last equation are leap-frog forms, which by themselves result in an unstable solution; the mere addition of the last (4.39) renders the system stable, and the solution is of $O(\delta t^2)$.

In their 1990 paper, Kimble & White [338] extended this idea to a higher-order formula. They replaced the second-order central difference with a fourth-order central difference using five points (and, as well, for the *pde* they were solving, a five-point second derivative with respect to x). This necessitates more asymmmetric formulae, both for the start and at the end. As with the former scheme, the five-point central difference approximation for y' (see Table A.1, form $y'_3(5)$), if used as a time-march, is unstable; and as with that scheme, the BDF form at the top end stabilises it and yields a high-order solution [141]. The formulae, again for a row of points $y_i, i = 1 \ldots N$ (given the boundary or initial value y_0) are the following.

For $i = 1$, a 5-point asymmetrical form, called $y'_2(5)$ in the Table, is applied to the point y_1:

$$\frac{-3y_0 - 10y_1 + 18y_2 - 6y_3 + y_4}{12\delta t} = f(y_1) \tag{4.41}$$

and from there on for all $i = 2, \ldots, N-2$, the central form $y'_3(5)$ is used,

$$\frac{y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}}{12\delta t} = f(y_i) \tag{4.42}$$

leaving the asymmetric form $y'_4(5)$ at $i = N-1$,

$$\frac{-y_{N-4} + 6y_{N-3} - 18y_{N-2} + 10y_{N-1} + 3y_N}{12\delta t} = f(y_{N-1}) \tag{4.43}$$

and finally, the BDF form $y'_5(5)$ at the top end,

$$\frac{3y_{N-4} - 16y_{N-3} + 36y_{N-2} - 48y_{N-1} + 25y_N}{12\delta t} = f(y_N)\ . \tag{4.44}$$

Again replacing the function terms $f(.)$ in these equations with our specific example function (4.3), $f(y) = -y$, and rearranging, this becomes the system

$$\begin{bmatrix}
(-10+12\delta t) & 18 & -6 & 1 & & & & \\
-8 & 12\delta t & 8 & -1 & & & & \\
1 & -8 & 12\delta t & 8 & -1 & & & \\
 & \ddots & & & & \ddots & & \\
 & & 1 & -8 & 12\delta t & 8 & -1 & \\
 & & -1 & 6 & -18 & (10+12\delta t) & 3 & \\
 & & 3 & -16 & 36 & -48 & (25+12\delta t) &
\end{bmatrix}
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-2} \\ y_{N-1} \\ y_N \end{bmatrix}
=
\begin{bmatrix} 3y_0 \\ -y_0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}\ . \tag{4.45}$$

This, as has been mentioned above, was not the way Kimble & White applied the method, but it works rather well and may well be more practical for *ode*s. In their application to *pde*s, the authors reduced the field to only a few points in both time and space, because otherwise the matrix to be solved, even when restricted to the pentadiagonal form, becomes very large.

4.10.1 Using KW as a Start for BDF

In the present author's view, the KW method might find its most useful application in providing high-order starting values for the BDF method, both for *ode*s and *pde*s, where this idea has been dubbed a "hyperimplicit" algorithm by Feldberg & Goldstein [236], presumably because it computes a number of time levels simultaneously. The idea was broadly explained above in Sect. 4.8.1 but can now be detailed. In order to start the BDF steps, the first few points need to be known; using k-point BDF, this means, strictly speaking, the initial value y_0 plus the next points up to y_{k-2}. When analysing this, however, it becomes clear that it is better to generate new points up to y_{k-1}, in order to get as high an error order as the BDF method itself. For example, to start a 5-point BDF simulation, we compute points $y_1 \ldots y_4$. To

get them, we can use KW. We restrict the discussion to the 5-point example. It has four unknowns, for which we must write four 5-point equations. It turns out that there are more possibilities than we require, and we can choose which four we use; all but the central one at y_2 are asymmmetrical, and so we choose a pure BDF form at y_4, and choose not to refer to y_0 (although we could do so). The equations are then, for the indicated i's, using again the 5-point forms in Table A.1 (and moving the divisor $12\delta t$ to the right-hand side)

$$\begin{aligned} -3y_0 - 10y_1 + 18y_2 - 6y_3 + y_4 &= 12\delta t f(y_1) \\ y_0 - 8y_1 + 8y_3 - y_4 &= 12\delta t f(y_2) \\ -y_0 + 6y_1 - 18y_2 + 10y_3 + 3y_4 &= 12\delta t f(y_3) \\ 3y_0 - 16y_1 + 36y_2 - 48y_3 + 25y_4 &= 12\delta t f(y_4) \end{aligned} \tag{4.46}$$

which, applied again to the example *ode* $y' = -y$, produces the system

$$\begin{bmatrix} (-10+12\delta t) & 18 & -6 & 1 \\ -8 & 12\delta t & 8 & -1 \\ 6 & -18 & (10+12\delta t) & 3 \\ -16 & 36 & -48 & (25+12\delta t) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 3y_0 \\ -y_0 \\ y_0 \\ -3y_0 \end{bmatrix} . \tag{4.47}$$

This works rather well with *ode*s, as also seen in Fig. 4.7. For use with *pde*s, however, it may be considered too much trouble to program, especially as there are easier options, for example extrapolation, which produce results that are just as good. Also, if BDF is nonetheless chosen, it was found in Sect. 4.8.1 and proved mathematically in Appendix B, that the simple start with a simple time correction produces about equally good results for much less effort.

4.11 Systems of *ode*s

All the techniques described above can also be applied to the numerical solution of systems of *ode*s, and here we are getting closer to what happens when we solve *pde*s, because in effect, one reduces them to *ode* systems when discretising them.

Instead of a single variable y, there is now a number n of variables, $y_1, y_2, \ldots, y_n$, represented by the vector $\boldsymbol{y}$. Each of these variables has its own differential equation, involving some function, on the function side, of the whole vector:

$$\begin{aligned} y_1' &= f_1(\boldsymbol{y}) \\ &\vdots \\ y_i' &= f_i(\boldsymbol{y}) \\ &\vdots \\ y_n' &= f_n(\boldsymbol{y}) \end{aligned} \tag{4.48}$$

(4.49)

where each f_i is some linear combination of the elements of the vector $\boldsymbol{y}$. We are concerned here only with linear systems; in those cases where a *pde* gives rise, upon discretisation, to a nonlinear system of *ode*s, tricks are normally used to avoid them, as will be seen in later chapters. The system is then conveniently written in vector-matrix form,

$$\boldsymbol{y'} = \boldsymbol{f}(\boldsymbol{y}) \tag{4.50}$$

or, since this is linear, as

$$\boldsymbol{y'} = \boldsymbol{A}\boldsymbol{y} \ , \tag{4.51}$$

where $\boldsymbol{A}$ is the matrix of coefficients in the n functions f_i. The system requires a set of boundary conditions (for example, initial values) for its solution, or

$$\boldsymbol{y'}(0) = \boldsymbol{y}_0 \ . \tag{4.52}$$

In principle, all the methods described above for single *ode*s can be used for the solution of such a system, when extended suitably. In the case of explicit methods such as Euler or RK, this is very simple to implement, whereas with implicit methods such as BI or the trapezium method, there are some choices to be made.

For brevity, the Euler method will be treated as a special case of RK, considered as RK1. The method is then to start by calculating a vector of k_1 values, one for each y element. Discretising directly from (4.51), this is

$$\boldsymbol{k_1} = \delta t \boldsymbol{A}\boldsymbol{y} \tag{4.53}$$

followed by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \boldsymbol{k_1} \tag{4.54}$$

for the Euler method, where $\boldsymbol{y}_{n+1}$ is the next value of the whole vector $\boldsymbol{y}$ in the iteration. The extension to RK is obvious. Note that the vector $\boldsymbol{k_1}$ (and, for higher RKn, the other $\boldsymbol{k_i}$ vectors) can be computed one element after the other, from known elements of $\boldsymbol{y}$, this being an explicit method.

Of implicit methods, two will be mentioned here, the first being BI. As outlined in Sect. 4.6, this involves equating the same time derivative used in all the methods, with the function on the right-hand side, referred to the next time interval. For the system, then,

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{\delta t} = \boldsymbol{A}\boldsymbol{y}_{n+1} \tag{4.55}$$

or

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \delta t \boldsymbol{A}\boldsymbol{y}_{n+1} \tag{4.56}$$

an implicit equation. The solution can be expressed as

$$\boldsymbol{y}_{n+1} = \left(\boldsymbol{I} - \delta t \boldsymbol{A}\right)^{-1} \boldsymbol{y}_n \tag{4.57}$$

in which $\boldsymbol{I}$ stands for the identity matrix. Similarly, the trapezium method starts with the discretisation

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{\delta t} = \left(\boldsymbol{A}\boldsymbol{y}_{n+1} + \boldsymbol{A}\boldsymbol{y}_n\right)/2 \tag{4.58}$$

leading finally to

$$\boldsymbol{y}_{n+1} = \left(\boldsymbol{I} - \tfrac{1}{2}\delta t \boldsymbol{A}\right)^{-1} \left(\boldsymbol{I} + \tfrac{1}{2}\delta t \boldsymbol{A}\right) \boldsymbol{y}_n \ . \tag{4.59}$$

These solutions are rather formal statements, and are rarely used as such, because the matrices involved are almost always either tridiagonal or pentadiagonal, making such direct solutions wasteful. It has been done in some cases [68,138], without any attempt at optimisation. It is possible to use solution methods that recognise the sparse nature of these systems and many professional program packages are available. One of these will be mentioned below. For methods for *pde*s corresponding to BI, trapezium and BDF, there are more efficient procedures for the solution, to be described in a later chapter.

In some cases, for example electrochemical *pde*s with derivative boundary conditions, the discretisation process for both the *pde* and the boundary conditions leads to a mix of a differential equation system and one or more plain algebraic equations. They might be, for example, equations of the form

$$\boldsymbol{f}(\boldsymbol{y}) = 0 \ . \tag{4.60}$$

The resulting system is called a set of **differential-algebraic equations** (**DAE**) and their solution is now a specialised field with its own texts [130, 286] and there is a package program, **DASSL** [441], for their solution. This can be of use in the present context, for example with the method of lines, which indeed often results in a **DAE** system. This is gone into in some detail in Chap. 9, in the context of Rosenbrock methods.

With most of the implicit methods to be described, however, the solution is found by specialised techniques that make the process efficient, and these will be described in their proper place.

4.12 Rosenbrock Methods

This section is left to the last because it pertains to systems of *ode*s (or **DAE**s), although Rosenbrock methods are a kind of Runge-Kutta method

(Sect. 4.5). In RK, a number of trial changes k_i are explicitly computed, and a weighted sum of them applied to the variable (or vector). As noted in that Section, such explicit RK methods can be highly accurate, but are not stable for all step-sizes, a limiting factor when applying them to diffusion problems. A better way is to use implicit Runge-Kutta formulae. With these, assuming, say, s trial k_i, the s equations contain expressions in all k_i (some of them perhaps left out, that is, with zero coefficients). This gives rise to a system of s equations in the vector $\boldsymbol{k}$, which can be troublesome. The advantage is that these implicit methods can lead to highly accurate, and stable, responses.

An alternative, called "semi-implicit methods" in such texts as [351], avoids the problems, and some of the variants are L-stable (see Chap. 14 for an explanation of this term), a desirable property. This was devised by Rosenbrock in 1962 [474]. There are two strong points about this set of formulae. One is that the constants in the implicit set of equations for the k's are chosen such that each k_i can be evaluated explicitly by easy rearrangement of each equation. The other is that the method lends itself ideally to nonlinear functions, not requiring iteration, because it is, in a sense, already built-in. This is explained below.

Consider the problem of a nonlinear *ode*

$$y' = f(t, y) \; . \tag{4.61}$$

We might wish to solve it using an implicit method, for example, BI (Sect. 4.6). Discretising (4.61) then gives

$$y_{n+1} - y_n = \delta t \, f(t + \delta t, y_{n+1}) \tag{4.62}$$

and the function on the right-hand side might not be known, nor might we be able to isolate the unknown, y_{n+1}, as was possible with the simple *ode*, $y' = -y$. The essence of Rosenbrock is now to take a single Taylor step, expanding $f(t + \delta t, y_{n+1})$ around $f(t, y_n)$ (and given that $y_{n+1} = y_n + k_1$),

$$f(t + \delta t, y_{n+1}) = f(t, y_n) + k_1 f_y(t, y_n) + \delta t f_t(t, y_n) \tag{4.63}$$

where the f_y denotes differentiation by y, and f_t that by t. We need to know these differentials, but this is always easy. Now (4.62) can be written in the more amenable form,

$$y_{n+1} - y_n = \delta t \big(f(t, y_n) + k_1 f_y(t, y_n) + \delta t f_t(t, y_n) \big) \; . \tag{4.64}$$

The right-hand side is in fact the expression for k_1, and contains it; hence the equation for k_1,

$$k_1 = \delta t \big(f(t, y_n) + k_1 f_y(t, y_n) + \delta t f_t(t, y_n) \big) \; , \tag{4.65}$$

is indeed implicit, but note that now k_1 can be isolated on the left-hand side, leaving only terms on the right-hand side that can be evaluated. We get

$$(1 - \delta t f_y(t, y_n))k_1 = \delta t f(t, y_n) + \delta t^2 f_t(t, y_n) \ , \tag{4.66}$$

easily solved for k_1. The formula might be called a one-stage Rosenbrock variant, and can in fact be used, although it is not highly accurate. Applied to our *ode*, $y' = -y$, for which $f_y = -1$ and $f_t = 0$, it results in the formula

$$y_{n+1} = \frac{y_n}{1 + \delta t} \tag{4.67}$$

which is seen to be identical with BI (in this case). Better than BI, however, (4.65) can be used for nonlinear equations, and also (including f_t) to *ode*s where t plays a role, the so-called nonautonomous *ode*s. This is an important point later, when applying Rosenbrock to diffusion problems, where time sometimes enters the equations through boundary conditions (for example LSV).

Before moving on to real Rosenbrock methods, consider again (4.66). The left-hand side contains a term in f_y; if we are dealing with a system of *ode*s, this is called the Jacobian of the system. It is often constant, evaluable in advance. It will be seen in Chap. 9 that unless the diffusion problem has nonlinear concentration terms (for example from higher-order homogeneous reactions), the Jacobian is constant. If not, it must be evaluated at every step.

There are several Rosenbrock variants, and a profusion of symbols used. Rosenbrock originally described a second-order variant, that is, with errors of $O(\delta t^2)$. It was for an autonomous *ode*, not involving t in the function on the right-hand side of (4.61), and the formula can be readily extended to the nonautonomous case (involving t) by a procedure described in [351] and [286], among others. Briefly, the procedure consists in adding the time variable to the vector y, and taking into account that $t' = 1$, and then expanding the formula appropriately. The formula given by Rosenbrock (and in [351]) then expands to that given in Appendix A. In this book, the profusion of symbols is reduced to one consistent set, as used in Hairer & Wanner [286], and this set is used exclusively in the table of constants in the Appendix. It is also the set adopted by Lang in his publications [345, 346, 347]. Lang, as will be noted, is the source of two new variants, at least one of them L-stable, and a modification of an existing one, as well as updated tables of coefficients, correct to 16 decimals in an Appendix in [347]. They are reproduced in the present Appendix A.

In general, a Rosenbrock method consists of a number s of stages. At each stage, a Runge-Kutta-type k_i value is calculated, from explicit rearrangement of implicit equations for these. At stage i, the equation is

$$\begin{aligned} k_i' = \ & \delta t \ f\Big(t + \alpha_i \delta t, y + \sum_{j=1}^{i-1} \alpha_{ij} k_j'\Big) \\ & + \delta t f_y(t, y) \sum_{j=1}^{i} \gamma_{ij} k_j' + \gamma_i \delta t^2 f_t(t, y) \ . \end{aligned} \tag{4.68}$$

The reason for writing k' is that this equation, applied to *ode* systems, has a small problem, in that the middle of the three terms on the right-hand side contains, in the sum, products of the form $f_y(t,y)\gamma_{ij}k'_j$, which for *ode* systems would mean multiplication of a Jacobian (the equivalent of $f_y(t,y)$) with the vectors $\boldsymbol{k}'_j$. Hairer and Wanner [286] show that by a transformation of the k' values into new k values,

$$k_i = \sum_{j=1}^{i} \gamma_{ij} k'_j \,, \tag{4.69}$$

a new equation using the k_i can be developed, avoiding this problem. Application of this transformation to (4.68) and rearranging, then leads to a new equation. We now assume that this concerns a system of *ode*s with vectors $\boldsymbol{y}$ and $\boldsymbol{k}$. Also, all γ_{ii} are conveniently chosen equal and are now simply called γ, and new constants γ_i, a_{ij} and c_{ij} appear. The final result is the explicit form,

$$(\boldsymbol{I} - \delta t\gamma \boldsymbol{f}_y)\boldsymbol{k}_i = \gamma \left(\delta t \boldsymbol{f} \left(t + \alpha_i \delta t, \boldsymbol{y} + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j \right) + \gamma_i \delta t^2 \boldsymbol{f}_t + \sum_{j=1}^{i-1} c_{ij} \boldsymbol{k}_j \right) \tag{4.70}$$

(we write $\boldsymbol{f}_y$ and $\boldsymbol{f}_t$ without their arguments for brevity). Note that in texts such as [286, 347], the equation presented is divided on both sides by $\gamma\delta t$. There are practical reasons for not doing this in the present context. The equation must be applied s times, according to the variant employed. Most variants seek to make the calculation convenient, by allowing some of the constants to be zero. A useful (and L-stable) second-order formula was described by Lang [347], called **ROS2**. A favourite third-order variant is **ROWDA3**, described by Roche [473] and later developed by Lang [346], making it more efficient. This is the variant favoured by Bieniasz, who introduced Rosenbrock methods to electrochemical digital simulation [100, 113] (using different symbols).

Having calculated the s $\boldsymbol{k}_i$ values (vectors), the solution is

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{i=1}^{s} m_i \boldsymbol{k}_i \,, \tag{4.71}$$

where the m_i are weighting factors, included in the tables of constants for each method.

4.12.1 Application to a Simple Example ODE

A simple example serves to illustrate the use of Rosenbrock, using the *ode*

$$y' = t + y \,; \qquad y(0) = 1 \tag{4.72}$$

which has the analytical solution [266]

$$y(t) = 2\exp(t) - t - 1\ . \tag{4.73}$$

This is of interest because it contains t, so that we must use both f_y and f_t, both equal to unity. Applying the ROS2 variant to this (see Appendix A for the coefficient values), (4.70) (now for a single variable y) translates to the two equations

$$\begin{aligned} k_1 &= \frac{\gamma\left(\delta t f(t,y) + \gamma_1 \delta t^2 f_t\right)}{1 - \gamma\delta t f_y} \\ &= \frac{\gamma\left(\delta t(t + y_n) + \gamma_1 \delta t^2\right)}{1 - \gamma\delta t} \end{aligned} \tag{4.74}$$

and

$$\begin{aligned} k_2 &= \frac{\gamma\left(\delta t f(t + \alpha_2\delta t, y + a_{21}k_1) + \gamma_2\delta t^2 + c_{21}k_1\right)}{(1 - \gamma\delta t f_y)} \\ &= \frac{\gamma\left(\delta t(t + \alpha_2\delta t + y + a_{21}k_1) + \gamma_2\delta t^2 + c_{21}k_1\right)}{(1 - \gamma\delta t)} \end{aligned} \tag{4.75}$$

containing the (now) known k_1. Then applying

$$y_{n+1} = y_n + m_1 k_1 + m_2 k_2 \tag{4.76}$$

yields the solution. This works out rather well, and tests show that errors are $O(\delta t^2)$.

4.12.2 Error Estimates

In publications providing Rosenbrock coefficients such as [100,347], there appear alternative coefficients, "hatted", such as $\hat{m}_1$ etc. These always provide another variant with an order smaller by one than the one used. The purpose of this is that the difference between the two forms provides an (over)estimate of the error. The practice is not followed in this book, as we are generally mainly interested in the error order. So these alternative, lower-order coefficients are not included in Appendix A.

5 The Explicit Method

The simplest method of simulating for *pde*s, and in particular for *ode*s, is the Euler method, in the present context usually called the "explicit method", or **EX** hereafter. It has many drawbacks (to be outlined) but it does have the advantage of simplicity of programming and if you are willing to let your computer do the hard work, it can yield adequate results in many cases. There are recent examples of the use of the method even for rather complex systems [324, 349] and a textbook on cyclic voltammetry [274] advocates the method (and provides a program in Pascal). One might thus choose the method as such, or choose to use it as a learning tool. The present author prefers the latter. Having learned how to use **EX** and aware of its drawbacks, one might be ready to learn something more advanced.

5.1 The Discretisation

The discussion will be restricted to the point method and to the one-dimensional case. We will now work in normalised variables, see Sect. 2.3. We then have concentration points $C_0, C_1, \ldots C_N, C_{N+1}$, at the locations $X = 0, H, \ldots NH, (N+1)H$, H being the interval in X, see Fig. 5.1. The end points at $X = 0$ and $X = (N+1)H$ are boundary points with concentrations C_0 and C_{N+1} respectively. It is the concentrations between these, that are subject to diffusional changes, as follows.

Fig. 5.1. Discrete sample point sequence

At any point with index i, that is at $X = iH$, the diffusion (1.1) is discretised on the left-hand side in the Euler manner (4.4, or in other words the forward difference formula 3.1) and on the right-hand side with the central three-point approximation (3.41), giving for the iteration going from time T to the next time $T + \delta T$,

$$\frac{C_i(T+\delta T) - C_i(T)}{\delta T} = \frac{1}{H^2}\left(C_{i-1}(T) - 2C_i(T) + C_{i+1}(T)\right) \ . \qquad (5.1)$$

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 73–83 (2005)
www.springerlink.com

The notation is now simplified by always assuming that we are at time T and are going to time $T + \delta T$ and writing $C(T + \delta T)$ as C'. The equation then rearranges to

$$C'_i = C_i + \lambda \left(C_{i-1} - 2C_i + C_{i+1} \right) \tag{5.2}$$

which is explicit for C'_i, the new concentration (noting that we have combined δT and H into $\lambda = \delta T / H^2$). This equation can be simply applied at all points. At the first and last points ($i = 1$ and N), the expression on the right contains a boundary term. For $i = 1$ we thus have

$$C'_1 = C_1 + \lambda \left(C_0 - 2C_1 + C_2 \right) \tag{5.3}$$

and for $i = N$,

$$C'_N = C_N + \lambda \left(C_{N-1} - 2C_N + C_{N+1} \right) \ . \tag{5.4}$$

The outer point C_{N+1} is normally equal to the bulk initial value, and thus equal to unity, since concentrations have been normalised by the bulk value (in cases involving more than one diffusing species, their respective initial bulk values, normalised by that of the chosen main species). The value of C_0 is a little more complicated to set. It depends on the experiment to be simulated, and for simplicity at this point the discussion will be postponed to a later section in this chapter. Assume that we know the value C_0. Then we need only go through all concentrations $C_1 \ldots C_N$, applying formula (5.2), and obtain the new row of values $C'_1 \ldots C'_N$.

5.2 Practicalities

In a given computer program, the section in which the above formula (5.2) appears, will normally be the shortest part of the program. Other parts of the program will read in the required parameters or print out the current where needed. This raises the question of which parameters to choose. Among these are the length of time assumed for the experiment to be simulated or, in our dimensionless terms, the number of units of the time used for normalisation, T_{max}. If the experiment is a step technique, this will mostly be unity. In the case of linear sweep voltammetry (where, due to the normalisation, time and potential have the same magnitude), it will be the number of potential units to be swept through (see Sect. 2.3). In either case, we must decide how many intervals (N_T) there are to be per time unit or (conversely) the interval length δT. Other parameters to be decided are the interval H and the number N of points along X. This is best done in terms of the largest X-value, which in turn is set, via (2.40) and (2.43), such that there are no diffusional changes beyond this point. In view of the properties of the error function and (2.43), a value of X_{max} given by

$$X_{max} = 6\sqrt{T_{max}} \tag{5.5}$$

is recommended. Next, we need the interval H or conversely, the number of points N along the X-axis. The recommendation here is to set this indirectly by means of the value of λ, since this value is known to affect the stability of the simulation critically. For the above formula, the largest usable value for it is 0.5. Having set its value, as well as that of either N_T (preferred) or δT, λ then sets N and H.

One also needs to think about at which points in time to output the current, being almost always the aim of the simulation. If this is to be plotted, it might be output at every new time into a plotting file. If one is testing a method, one might only output the current at a selected number of intervals. The present author finds it most convenient to output it at expanding time intervals, as this always gives a compact output list. The current itself can be calculated from the concentration profile by a number of approximations, depending upon how many points one takes. In Chap. 3 (page 39), the function $\mathcal{G}$ was defined to evaluate the gradient G. The simplest is the two-point forward difference formula (3.1) which is $\mathcal{G}(C, 2, H)$ or

$$G \approx \frac{C_1 - C_0}{H} \tag{5.6}$$

preferred by many (G here is the dimensionless current or gradient). It is, however, as described in Chap. 3, first-order with respect to H and, with very little effort, one can do much better than that. Since the three-point approximation to the second derivative, the right-hand side of the diffusion equation as shown above in (5.2), is second-order with respect to H, there are good arguments [100] for using a second order formula for G, the form $y_1'(3)$ in Table A.1, or

$$G \approx \frac{-3C_0 + 4C_1 - C_2}{2H} \,. \tag{5.7}$$

The present author prefers to use an even larger number of points, having established a function subroutine (see the collection discussed in Appendix C) `GOFUNC`, which implements the function $\mathcal{G}(C, n, H)$ and allows up to seven points ($n = 7$). One can then be sure that there are no significant error terms from the approximation to G, no matter which method is used. Furthermore, although the calculation does take a little longer than the simple two-point one, this is never a lot compared with the simulation iteration itself, so it does not matter.

Eyres et al. [225] used a three-point flux approximation for heat flow simulations; the earliest electrochemist to use simulation was Randles [460] and he also used a three-point current. Amatore and Savéant [50] used a six-point approximation, as did Bellamy et al. [85]. The latter authors also inverted the six-point formula to calculate C_0, in the manner of `COFUNC` discussed in Appendix C.

5.3 Chronoamperometry and -Potentiometry

The program `COTTEX` (see Appendix C) is an example of a simulation of the Cottrell experiment, using method EX. In this simplest of experiments, the boundary concentration C_0 is held at zero from $T = 0$ onwards (see Sect. 2.4.1). Boundary conditions will be dealt with in detail in the next chapter, but here it is mentioned only that this condition is the Dirichlet boundary condition, in which, in general, the value of C_0 is given. The other extreme condition is the Neumann condition, in which the gradient G is given (a derivative boundary condition). An example of this is given by chronopotentiometry, where a constant current is imposed on the electrode. There is an important point to make here in this regard. When stepping forward in time, one starts with a known array of concentration values C and uses them explicitly (in this case of method EX) to compute the new row, C'. Thus, since the old boundary values, C_0 and C_{N+1} also go into the (5.3) and (5.4), they must be available, and they must be appropriate to the other values. Therefore, in chronopotentiometry, when a given C-row has been computed, that value of C_0 must be computed which yields the correct gradient G, the boundary condition. This can be done readily by inverting the approximation for G. For the two-point approximation (5.6) this becomes

$$C_0 = C_1 - GH \tag{5.8}$$

while for the three-point G as in (5.7), it is

$$C_0 = \frac{1}{3}\left(4C_1 - C_2 - 2GH\right) \ . \tag{5.9}$$

In general, for n points, it becomes

$$C_0 = \frac{1}{\beta_0}\left(GH - \sum_{i=1}^{n-1} \beta_i C_i\right) \ . \tag{5.10}$$

This last general equation is implemented in the function `COFUNC`, described in Appendix C. The formula is applied in the seven-point form in the example program `CHRONOEX` described in Appendix C, simulating chronopotentiometry. It must be applied before every new iteration, in order for the C_0 value to be in line with the other C values. In this program, the current is constant and it is the value of C_0 which is displayed and this should go to zero at $T = 1$ (Sect. 2.4.2). A more appropriate display might be the electrode potential, which is always the measured quantity, but this will be dealt with together with the more detailed discussion of boundary conditions in Chap. 6.

A final practical point is the following. When computing the new row C', it must be computed from the old row. Therefore in a program, we cannot replace each C_i with the new value in the array immediately, for each i, because the neighbouring C_{i-1} has just been changed to C'_{i-1}. One can either

declare two separate arrays, which might be considered slightly wasteful of space, or use the small trick seen in the example programs (Appendix C), where both `COTTEX` and `CHRONOEX` use a trio of scalar points that have the current three values C_{i-1}, C_i, C_{i+1} that are needed. In this way, the new value C_i' can replace the old one in the array, which is preserved in the scalar point trio. This must of course be shifted along at the bottom of the loop, and the point C_{i+1} picked up at every new loop restart. This device is possibly less important these days, since computers have more memory and an extra array can be easily accommodated.

5.4 Homogeneous Chemical Reactions (*hcr*)

One of the simplest examples of a homogeneous chemical reaction (*hcr*) is the Reinert-Berg system [464], in which an electroactive species is generated, for example by means of a light flash, and then reduced as a Cottrell system, while the species decays chemically with a first-order reaction. The reactions are then

$$\begin{aligned} A + e^- &\rightarrow B \\ A &\rightarrow prod \end{aligned} \tag{5.11}$$

with k the rate constant of the second reaction, the *hcr*. This gives rise to the single governing equation, for substance A

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} - KC \tag{5.12}$$

where everything, including the rate constant ($K = k\tau$) as described in Sect. 2.4, (2.64), has been normalised. The analytical solution of this equation with Cottrell boundary conditions is given in Chap. 2, Sect. 2.4, (2.67) and (2.68). This system is very simple to discretise in the EX manner; the chemical term is added to the equations above, (5.2)–(5.4). For the ith concentration (5.2) this becomes

$$C_i' = C_i + \lambda\left(C_{i-1} - 2C_i + C_{i+1}\right) - K\delta T C_i \; . \tag{5.13}$$

If several species are involved (in this case there is the product *prod*, but we are not interested in it), the equations are extended in an obvious manner, apart from some tricks to be seen in a later chapter in connection with implicit methods. This is one of the attractive aspects of method EX. If the *hcr* is second order, there will be a term in C_i^2 in the discrete equation, and it will present no problem in the discretisation step [146].

There are, however, several problems here. The first is that in writing the governing equation in discrete form as above, we are in effect uncoupling diffusional changes from chemical changes. Numerically, they appear to take

place independently of each other, whereas in fact, they interact during the time interval. This leads to inaccuracies, and Nielsen et al. [418] proposed using the (explicit) Runge-Kutta method (RK) to overcome this problem, if the chemical changes during a single time interval amount to a few percent of the concentration itself or, in mathematical terms, if the quantity $K\delta T$ in (5.13) exceeds 0.01 or so. In [418] the method adopted was to use RK for the chemical reaction only, following earlier suggestions in the literature [247], whereas in [135] the method is applied to the whole equation, to be described in Chap. 9. This turns out, in either case, to give only a modest improvement in efficiency, which can however be improved a little, see the 5-point method, Chap. 9.

The above manner of computing the two components, that is, diffusional and chemical changes separately, is sometimes called the parallel method. Another way to improve efficiency slightly is to use instead what has been called [418] the sequential method. It was intuitively applied at first, without any real justification other than that it gave better results. The method consists of calculating the diffusional change first, augmenting the concentrations by these amounts, and then to apply the chemical reaction to these augmented values. As pointed out in [418], this gives rather good results, but it was not clear at that time why this should be so. Feldberg in fact used this method and describes it in his original monograph [229]. It turns out that, by coincidence (of which we have several in digital simulation), the parallel method does have a mathematical justification and is consistent with the model equations that the discrete expressions approximate. A mathematical proof of this consistency was given in 1991 [485] and is reproduced in Appendix B. The improvement is not great, however, and other methods were sought.

If one is computing the two changes separately (the parallel method), and given that the chemical reaction itself is usually tractable analytically, this component need not be simulated. For a first-order reaction as seen in (5.11) and (5.12), the last part in (5.12) has the general solution for a first-order reaction,

$$C(T) = C(0)\exp(-KT) \tag{5.14}$$

which becomes, over the interval δT, for the concentration C_i

$$C_i(T + \delta T) = C_i(T)\exp(-K\delta T) \tag{5.15}$$

and if $K\delta T$ is not large, this converges to

$$C_i(T + \delta T) = C_i(T)(1 - K\delta T) \tag{5.16}$$

which is indeed the chemical component in (5.13). Feldberg and Auerbach used the analytical method [234] in 1964, as did Flanagan and Marcoux [246] in 1973 and Amatore and Savéant in 1979. It has since then given way to other, better methods, due to the recognition that it is not justified

to assume that diffusional and chemical changes are separate. One of these better methods is the explicit Runge-Kutta method (RK) applied to the whole discrete equation set (5.12), and will be described in Chap. 9.

5.4.1 The Reaction Layer

All the above methods, when *hcr*s are present, have one very serious drawback: many *hcr*s give rise to a compact reaction layer, as described in Chap. 2. The above Reinert-Berg reaction does not, but the **EC** reaction,

$$\begin{aligned} A + e^- &\rightarrow B \\ B &\rightarrow C \end{aligned} \tag{5.17}$$

does. It turns out that the concentration profile for B extends less far into the solution than that of A, which follows the normal rules. Figure 5.2 illustrates this for the above mechanism, where concentration profiles for all three species are shown for the Cottrell experiment run on this system, at (dimensionless) unity time, and a dimensionless rate constant $K = 10$. The profile for species A is unchanged, that is, it is the same as if there were no following reaction, and shows the normal Nernst diffusion layer thickness δ. Species B, however, is confined to a narrower region and its interface concentration is smaller than it would be without the following reaction; species C contains the deficit in B. The problem here is that, if the rate constant is large, the reaction layer is very thin and in order to be able to approximate such a concentration profile, close spacing of points is required, increasing computer time. There are ways to overcome this, but the EX method with equally spaced intervals is not one of them.

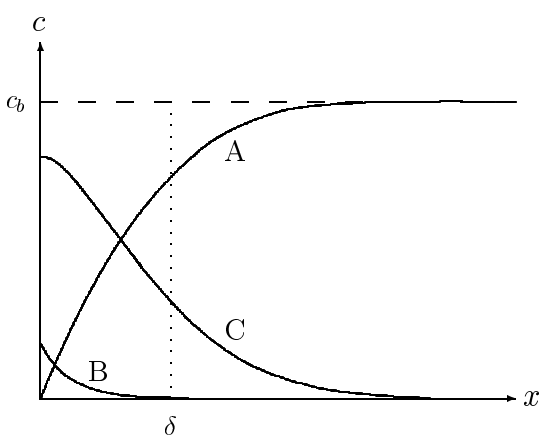


Fig. 5.2. Concentration profiles with reaction layers

5.5 Linear Sweep Voltammetry

One of the main uses of digital simulation – for some workers, the only application – is for linear sweep (LSV) or cyclic voltammetry (CV). This is more demanding than simulation of step methods, for which the simulation usually spans one observation time unit, whereas in LSV or CV, the characteristic time τ used to normalise time with is the time taken to sweep through one dimensionless potential unit (see Sect. 2.4.3) and typically, a sweep traverses around 24 of these units and a cyclic voltammogram twice that many. Thus, the explicit method is not very suitable, requiring rather many steps per unit, but will serve as a simple introduction. Also, the groundwork for the handling of boundary conditions for multispecies simulations is laid here.

The sequence of steps in a CV simulation program is as follows. A simple two-species reaction

$$A + e^- \leftrightarrow B \tag{5.18}$$

is our example, assumed to be quasireversible with dimensionless heterogeneous standard rate constant K_0:

1. Read in starting potential p_{start} and reversal potential p_{rev} (both in dimensionless potential units), n_{Tper}, the number of time intervals per potential unit swept, and λ, the simulation parameter, and K_0, the dimensionless heterogeneous rate constant.
2. Calculate some numbers derived from these inputs, such as N, the number of points in space (see below) and H, the interval along X and the total number of time steps n_T, as well as the time interval δt. Initialise arrays etc.
3. Open the required output files.
4. Set the potential step δp to $-\delta t$, and the current potential p to p_{start}; that is, the sweep starts in the negative direction.
5. Enter the loop, each time increasing the potential by δp, and computing the new concentrations. When these have been calculated, apply the boundary conditions for the current potential, to get the boundary values at $X = 0$; write out the potential and current into the file (perhaps only if there has been a change in current greater than some set value, to reduce the volume of output). If half the total number of time steps n_T has been done, flip the sign of δp, so that the next half will go in the reverse direction, for CV.
6. During the loop, monitor the current to detect when it passes through a peak (negative sweep) or a trough (return sweep), and keep these values and the potential where they occurred.
7. Finish up by writing out the required numbers such as maximum and minimum currents and potentials.

Some remarks on the above are in order. In the example program `CV_EX` mentioned in Appendix C, a quasiversible reaction is indeed assumed, but if

K_0 exceeds the value 1000, the boundary conditions are taken to be those for a reversible reaction. How these two different boundary conditions are applied to calculate the concentrations $C_{A,0}$ and $C_{B,0}$ is described below. Note that before new concentrations are to be computed, all old concentrations, including the boundary values, must be known. When a new potential is stepped to, it comes into effect only after the concentrations are renewed, after which C_0 is calculated. This might be thought of as less than satisfactory, but it is consistent with the explicit method. In Chaps. 8 and 9, more satisfactory methods will be presented.

Regarding the number N of points in space, the rule shown in (5.5) is used; the total time T_{max} here is equal to the total number of potential units swept through. If we, for example, set p_{start} equal to 12 and p_{rev} equal to –12, then $T_{max} = 48$. One could in principle save a little computing time by recognising the fact that after a given number n_s of steps taken, only that many concentrations can have changed, due to the way changes propagate through the concentration profiles in the explicit method, so while $n_s < N$, one need only recompute n_s points; but this is a small saving in computing time and is not worth the effort, or the risk of introducing a program error in the process.

Monitoring for peak and trough currents is seen in the example program `EX_CV`, by the device that a trio of current values G_1, G_2, G_3 is always kept (G_3 being the most recently computed value), and a check is made whether G_2 is maximum or minimium. If it is, the true peak or trough is computed, using the routine `MINMAX` described in Appendix C, which uses a parabolic fit to detect the values, as well as the position. This is converted to potential units in the program. There is a small device to prevent spurious peak/trough detections, by means of the restriction that only those taking place within the potential range $-2 < p < 2$ are accepted.

The number of steps for a complete CV simulation will generally be quite large, especially for the explicit method, where one must set the δp values down around at most 0.01, so that if currents are output for later plotting at each iteration, the file becomes unnecessarily large. For this reason, `EX_CV` checks that there has been a minimum change in current since the last, before writing the current out. Setting this value to 0.001 gave a reasonably reduced number of outputs in some test runs (1435, out of a total of 48000, with δp set to 0.001).

5.5.1 Boundary Condition Handling

In the example program `EX_CV`, as mentioned, two kinds of boundary conditions are accommodated: those for a quasireversible reaction, and for a fully reversible reaction. The division is made on the basis of the dimensionless heterogeneous rate constant K_0; if it exceeds 1000, the reaction is considered reversible.

For the quasireversible case, the procedure is as follows. At a given stage in the simulation, assume that the two concentration profiles, $C_{A,i}$ and $C_{B,i}$, with $i = 1, 2, \ldots, N$, have been calculated and that the potential is p. The dimensionless form of the Butler-Volmer equation applies (2.30) and provides the concentration gradient G_A, proportional to the current:

$$G_A = K_f C_{A,0} - K_b C_{B,0} \tag{5.19}$$

where the two constants, the forward and backward rate constants, are as given previously (2.31), functions of the potential. The left-hand side of this equation can be discretised as the n-point current approximation, leading to

$$\frac{1}{H} \sum_{i=0}^{n-1} \beta_i C_{A,i} = \mathcal{G}(C_A, n, H) = K_f C_{A,0} - K_b C_{B,0} \tag{5.20}$$

which can be dissected as

$$\frac{1}{H} \left(\beta_0 C_{A,0} + \sum_{i=1}^{n-1} \beta_i C_{A,i} \right) = K_f C_{A,0} - K_b C_{B,0} \, . \tag{5.21}$$

There are two unknowns, so we need one more equation. This comes from the fact that the flux of substance A at the electrode must be equal and opposite to that of substance B. If we assume equal diffusion coefficients for the moment, this means

$$G_A + G_B = 0 \tag{5.22}$$

and employing the current approximations for both species and dissecting as above, this and the former (5.21) rearranged, gives us the two-unknowns system

$$\begin{bmatrix} (K_f H - \beta_0) & -K_b H \\ \beta_0 & \beta_0 \end{bmatrix} \begin{bmatrix} C_{A,0} \\ C_{B,0} \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{i=1}^{n-1} \beta_i C_{A,i} \\ \displaystyle -\sum_{i=1}^{n-1} \beta_i C_{A,i} - \sum_{i=1}^{n-1} \beta_i C_{B,i} \end{bmatrix} \tag{5.23}$$

which is readily solved for the two boundary values. The sums can be obtained using the function `GOFUNC` discussed in Appendix C and a small trick. The function requires a number of C_i values, including the (unknown) C_0. However, since we are calculating it and do not need the old value, we can afford to set it to zero, so that the function represents the sum for $i = 1, 2, \ldots, n-1$, leaving out the zeroth element, as required in the above equations. This is made use of in the example program.

For the reversible case, the Nernst equation applies instead of the Butler-Volmer equation, that is, in dimensionless terms as in (2.32), rewritten as

$$C_{A,0} - e^p C_{B,0} = 0 \tag{5.24}$$

paired again with the flux equality condition (5.22). This gives the system in two unknowns

$$\begin{bmatrix} 1 & -e^p \\ \beta_0 & \beta_0 \end{bmatrix} \begin{bmatrix} C_{A,0} \\ C_{B,0} \end{bmatrix} = \begin{bmatrix} 0 \\ -\sum_{i=1}^{n-1} \beta_i C_{A,i} - \sum_{i=1}^{n-1} \beta_i C_{B,i} \end{bmatrix} \tag{5.25}$$

which is even easier to render as an explicit expression for $C_{A,0}$ and thereby for $C_{A,0}$ from (5.24).

More will be said about boundary conditions in Chap. 6.

6 Boundary Conditions

In this chapter, boundary conditions and how to handle them in simulations are described. Of necessity, some material here overlaps with that in other chapters, especially Chaps. 8 and 9; but this cannot be avoided.

Adsorption kinetics has its own boundary conditions and is treated entirely separately in Chap. 10.

6.1 Classification of Boundary Conditions

In the world of numerical analysis, one distinguishes formally between three kinds of boundary conditions [283, 528]: the Dirichlet, Neumann (derivative) and Robin (mixed) conditions; they are also sometimes called [283, 350] the first, second and third kind, respectively. In electrochemistry, we normally have to do with derivative boundary conditions, except in the case of the Cottrell experiment, that is, a jump to a potential where the concentration is forced to zero at the electrode (or, formally, to a constant value different from the initial bulk value). This is pure Dirichlet only for a single species simulation because if other species are involved, the flux condition must be applied, and it involves derivatives. Therefore, in what follows below, we briefly treat the single species case, which includes the Cottrell (Dirichlet) condition as well as derivative conditions, and then the two-species case, which always, at least in part, has derivative conditions. In a later section in this chapter, a mathematical formalism is described that includes all possible boundary conditions for a single species and can be useful in some more fundamental investigations.

In this chapter, the current approximation function $\mathcal{G}$, defined in Chap. 3, (3.25), will be used extensively. Note also that since this function is a linear combination of the array argument (for example, C as in $\mathcal{G}(C, n, H)$), the function of a weighted sum of two arrays, such as the arrays $\boldsymbol{u}$ and $\boldsymbol{v}$ (to be met later), the following holds (a being some scalar factor):

$$\mathcal{G}(\boldsymbol{u} + a\boldsymbol{v}, n, H) = \mathcal{G}(\boldsymbol{u}, n, H) + a\mathcal{G}(\boldsymbol{v}, n, H) \; . \tag{6.1}$$

This will prove useful in connection with the "**u-v**" device, see below.

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 85–102 (2005)
www.springerlink.com

6.2 Single Species: The u-v Device

If the simulation only involves a single substance (species), the situation is relatively simple, and this is the starting point. Some of this has already been described in Chap. 5 but will be repeated here, more generally, and with reference to implicit methods, not yet described. Recall the convention that a concentration denoted as C_i is a "present" value at the (spatial) index i, that is, a known concentration at time T, whereas C'_i denotes a value, yet to be calculated, at time $T + \delta T$.

6.2.1 Dirichlet Condition

Here the value of the boundary concentration is specified. A familiar example in the present context is the outer boundary, beyond the diffusion space, where the concentration usually remains at the initial bulk value during the whole period over which the simulation is carried out. This also applies to the case of the Reinert-Berg mechanism (page 20), in which the bulk concentration itself changes with time, but we know the bulk value at any time, because chemical reaction kinetics, uncomplicated by transport effects, is well understood. In such cases, we can set a given bulk concentration, albeit time-varying. Another familiar example arises from the Cottrell experiment, in which the concentration at the electrode, C_0, is set to zero. This is a particular case of that concentration being set to a definite value, not necessarily zero.

6.2.2 Derivative Boundary Conditions

For a single species, there are only two cases of interest, arising from the two kinds of experiments in which either the current is controlled or the potential is controlled, and the reaction is irreversible (if it is not irreversible, two species must be considered). These two cases can serve as a kind of tutorial for the more complex two-species systems.

The chronopotentiometry (controlled current) case has already been described for the **EX** method, where one simply finds a C_0 value that fits the known gradient G and the concentration points already established, as shown in (5.10). The situation is not quite so simple for implicit methods, and we introduce here both a preview of these, and the **u-v** device, which will be used extensively.

As will be seen in Chap. 8, implicit methods all lead (for the normal 3-point approximation of the term $\partial^2 C/\partial X^2$) to a system of N equations, each with three unknowns. Generally, this can be written in the form, for the i^{th} equation out of the N,

$$C'_{i-1} + a_1(i)C'_i + a_2(i)C'_{i+1} = b_i \tag{6.2}$$

where the coefficients $a_1(i)$ and $a_2(i)$ depend on the particular implicit method employed, and the b_i term is some weighted sum of known concentrations, again depending on the method. Because of the fact that the last equation, where $i = N$, includes the bulk value C'_{N+1} which is known, it is possible to reduce the set of equations recursively to one in which each equation has two unknowns, to be outlined in Chap. 8, going backwards from the outer value, and ending in the new equation system:

$$\begin{aligned} C'_0 + a'_1 C'_1 &= b'_1 \\ C'_1 + a'_2 C'_2 &= b'_2 \\ C'_2 + a'_3 C'_3 &= b'_3 \\ &\dots \end{aligned} \quad . \tag{6.3}$$

The details of how to get from the system (6.2) to (6.3) are described in Chap. 8. We can use (6.3) to solve for all C'_i, if we know the boundary value C'_0. In the case of the Cottrell system, we do know it; it is zero, thus giving us C'_1 directly, and then C'_2, etc. This is essentially the Thomas algorithm (see Chap. 8).

If there is a derivative boundary condition, things are a little more complicated. There are two kinds of cases. The first of these arises with controlled current, where we know the gradient G, as already seen in Chap. 5. Here, however, we cannot simply calculate C'_0, because we do not yet know the other concentrations. One way to handle this is to add an expression for the boundary condition to a few equations out of (6.3) and to solve. A simple example is to use the 2-point G-approximation in the case, for example, of controlled current (G), and the first equation from (6.3)

$$\begin{aligned} C'_1 - C'_0 &= GH \\ C'_0 + a'_1 C'_1 &= b'_1 \end{aligned} \tag{6.4}$$

and to solve for C'_0 (and C'_1), giving

$$C'_0 = \frac{b'_1 - a'_1 GH}{1 + a'_1} \quad . \tag{6.5}$$

This is convenient for the simple 2-point approximation but if $n > 2$, more equations out of (6.3) are needed, and the solution is less straightforward.

Another, more convenient way is the **u-v** device. We establish a relation between the concentrations C'_i and C'_0, in order to obtain the extra information needed to solve for C'_0. Taking the first equation in (6.3), we rewrite it explicitly for C'_1:

$$C'_1 = b'_1/a'_1 - C'_0/a'_1 \tag{6.6}$$

or as a linear function of C'_0,

$$C'_1 = u_1 + v_1 C'_0 \tag{6.7}$$

where

$$u_1 = b'_1/a'_1 \; ; \qquad v_1 = -1/a'_1 \; . \tag{6.8}$$

Equation 2 of the system (6.3) is then reorganised explicitly for C'_2, giving

$$C'_2 = b'_2/a'_2 - C'_1/a'_2 \tag{6.9}$$

and we substitute for C'_1 from (6.7), getting

$$C'_2 = b'_2/a'_2 - (u_1 + v_1 C'_0)/a'_2 \tag{6.10}$$

which again can be expressed as a linear expression in C'_0,

$$C'_2 = u_2 + v_2 C'_0 \tag{6.11}$$

where now

$$u_2 = (b'_1 - u'_1)/a'_1 \; ; \qquad v_2 = -v_1/a'_1 \; . \tag{6.12}$$

This can be continued and we can recursively express all C'_i as linear functions of C'_0. For the i^{th} concentration,

$$C'_i = u_i + v_i C'_0 \tag{6.13}$$

and the coefficients are given by the recursive expressions

$$u_i = (b'_i - u_{i-1})/a'_i \; ; \qquad v_i = -v_{i-1}/a'_i \; . \tag{6.14}$$

This is easily programmed as a loop process. If we want to avoid a special expression for u_1 and v_1, there is a trick: we start formally with a tautological equation,

$$C'_0 = u_0 + v_0 C'_0 \tag{6.15}$$

in which, obviously, $u_0 = 0$ and $v_0 = 1$. Then the loop process, applying (6.14) for $i = 1, 2, \ldots$, can be set running. We do not need many iterations – in fact, $n-1$ are sufficient, n being the number of concentrations used in the approximation for G. This is

$$G = \mathcal{G}(\boldsymbol{C}', n, H) \tag{6.16}$$

and substituting for all C'_i from (6.13), noting (6.1) and defining the vectors (arrays) $\boldsymbol{u} \equiv [u_0 \; u_1 \ldots u_{n-1}]^T$ and $\boldsymbol{v} \equiv [v_0 \; v_1 \ldots v_{n-1}]^T$, we get

$$G = \mathcal{G}(\boldsymbol{u}, n, H) + C'_0 \, \mathcal{G}(\boldsymbol{v}, n, H) \tag{6.17}$$

or multiplying by H,

$$GH = \mathcal{G}(\boldsymbol{u}, n, 1) + C'_0 \, \mathcal{G}(\boldsymbol{v}, n, 1) \tag{6.18}$$

which can now be rearranged to the explicit equation for C'_0,

$$C_0' = \frac{GH - \mathcal{G}(\boldsymbol{u}, n, 1)}{\mathcal{G}(\boldsymbol{v}, n, 1)} \tag{6.19}$$

yielding C_0'. It is now also seen that the little trick (6.15) has another advantage, enabling the use of the sums for $i = 0, 1, \ldots, n-1$, which our function $\mathcal{G}$ and thus procedure `GOFUNC` evaluates. `GOFUNC(u, n, 1.0_dbl)` must be called as shown and `u` and `v` are arrays with bounds `(0:n-1)`.

Formally, the above process is equivalent to (6.4), extended for any n and solving that system. The **u-v** device is a more efficient way of solving it than any linear equation solver that might otherwise have been used, as n becomes larger. The **u-v** device will be extensively used in this book, even with implicit methods for coupled equation systems, where we must solve for a number of concentration profiles (see below). There are practitioners who believe that $n = 2$, that is the two-point G-approximation, is good enough. This is justified in cases where H is very small, as it often is, at least near the electrode, when unequal intervals are used (see Chap. 9). In that case, one can simply use (6.5).

Another case, if we have just one species, besides controlled current, is the irreversible, controlled potential case. The gradient G is then given by half of the Butler-Volmer equation and for the as yet unknown concentrations and the potential p' at the new time, this is

$$G' = K_f C_0' \tag{6.20}$$

(where we write G', since the gradient is now time dependent and we are referring to the new time level) with K_f given by

$$K_f = K_0 \exp(-\alpha p') \; . \tag{6.21}$$

G' itself must then be replaced by the right-hand side of (6.16) and the resulting equation,

$$\mathcal{G}(C', n, H) = K_f C_0' \tag{6.22}$$

decomposed by applying (6.13) to the left-hand side. After some rearrangement, this yields the solution,

$$C_0' = \frac{\mathcal{G}(\boldsymbol{u}, n, 1)}{HK_f - \mathcal{G}(\boldsymbol{v}, n, 1)} \; . \tag{6.23}$$

Again, if a two-point approximation is used, this simplifies to

$$C_0' = \frac{b_1'}{1 + a_1'(1 + HK_f)} \; . \tag{6.24}$$

Lastly, in Chap. 9, Sect. 9.2.7, an improvement in the above is described, based on Hermitian schemes for a better gradient approximation.

6.3 Two Species

In Chap. 5, the two-species cases were described for the explicit method. Here we add those for the implicit case. Both Dirichlet and derivative boundary conditions are of interest, the latter both with controlled current or quasireversible and systems under controlled potential.

When two species are involved, they may have different diffusion coefficients. Here it will be assumed that the two species might be two out of more than two species in a given mechanism, and that normalisation is referred to some species other than these two. Therefore both their diffusion coefficients need to be normalised. Let the two species be called O and R, and the reference species be called A. Then the normalisations are

$$d_O = D_O/D_A\ , \qquad d_R = D_R/D_A \tag{6.25}$$

and, of course, the concentrations are normalised as usual by the initial bulk concentration of the reference species A.

Often, for convenience, diffusion coefficients of all species in a mechanism are assumed equal, however unrealistic this probably is. If the reader wants to assume this, the d's in what follows can simply be set equal to unity. A recent paper [504] underlines the point, finding significant effects of unequal diffusion coefficients, and Pedersen et al. [438] in fact measured differences of about 20% between the diffusion coefficients of some organic compounds and the radical anions formed by their reduction. Such differences may well be significant in a simulation.

We consider here two species connected by the reduction reaction

$$O + e^- \rightarrow R\ . \tag{6.26}$$

We begin with the simpler case of the two species not being coupled, that is, each of their discrete equations contains only terms from one of the species. The coupled case is given below, being rather more complicated. There is of course coupling of the two species by the boundary conditions.

As with the single-species case above, we anticipate the treatment given in Chap. 8 for implicit methods. At each point i in space, there is an equation like (6.2) for each species:

$$\begin{aligned} C'_{O,i-1} + a_{O,1}(i)C'_{O,i} + a_{O,2}(i)C'_{O,i+1} &= b_{O,i} \\ C'_{R,i-1} + a_{R,1}(i)C'_{R,i} + a_{R,2}(i)C'_{R,i+1} &= b_{R,i} \end{aligned} \tag{6.27}$$

where the coefficients may be different (often some of them at least are common to the two). The two equations are of the same form as (6.2), except that there are now twice as many. Again, the bulk concentrations will be known and can be used to reduce the whole set to a new set in which each equation has only two unknowns, and we write out the first few of these:

$$\begin{aligned} C'_{O,0} + a'_{O,1}C'_{O,1} &= b'_{O,1} \\ C'_{R,0} + a'_{R,1}C'_{R,1} &= b'_{R,1} \\ C'_{O,1} + a'_{O,2}C'_{O,2} &= b'_{O,2} \\ C'_{R,1} + a'_{R,2}C'_{R,2} &= b'_{R,2}\,. \\ C'_{O,2} + a'_{O,3}C'_{O,3} &= b'_{O,3} \\ C'_{R,2} + a'_{R,3}C'_{R,3} &= b'_{R,3} \\ &\dots \end{aligned} \tag{6.28}$$

It might have been clearer to write these two systems separately but it was decided to mix them in the above manner, as this serves a certain purpose later, with coupled systems.

The **u-v** device can now be applied as before, the only complication being that there will be two sets of $\boldsymbol{u}$'s and $\boldsymbol{v}$'s; the treatment is identical to the above one and results in the two equations

$$C'_{Z,i} = u_{Z,i} + v_{Z,i}C'_{Z,0} \tag{6.29}$$

in which Z can be either O or R. The equations that generate the coefficients are the same as (6.14).

It is now possible to bring in the particular boundary conditions, starting with the Cottrell case, which is the simplest. For all mechanisms and boundary conditions, we require two equations involving the two unknown boundary concentrations. As with all cases, one of these is the flux condition,

$$f_O + f_R = 0 \tag{6.30}$$

mentioned in Chap. 5, Sect. 5.5.1. We must take the possibly different diffusion coefficients into account, since it is the fluxes, not the concentration gradients, that must be equal and opposite:

$$D_O G_O + D_R G_R = 0 \tag{6.31}$$

or, applying the normalisations of the diffusion coefficients,

$$d_O G_O + d_R G_R = 0\,. \tag{6.32}$$

This then becomes, for the usual n-point G-approximation,

$$d_O \mathcal{G}(\boldsymbol{C}'_O, n, H) + d_R \mathcal{G}(\boldsymbol{C}'_R, n, H) = 0 \tag{6.33}$$

(where $\boldsymbol{C}'_O$ stands for the vector of the values $\boldsymbol{C}'_{O,0}\dots$ and similarly for $\boldsymbol{C}'_R$) and substituting (6.14) for $C'_{O,i}$ and $C'_{R,i}$, and rearranging, this gives the equation

$$d_O \mathcal{G}(\boldsymbol{v}_O, n, 1)C'_{O,0} + d_R \mathcal{G}(\boldsymbol{v}_R, n, 1)C'_{R,0} = -d_O \mathcal{G}(\boldsymbol{u}_O, n, 1) - d_R \mathcal{G}(\boldsymbol{u}_R, n, 1) \tag{6.34}$$

where the H has been divided out, but this is a matter of preference. The equation will be written in a briefer form,

$$d_O\mathcal{G}(\boldsymbol{v}_O)C'_{O,0} + d_R\mathcal{G}(\boldsymbol{v}_R)C'_{R,0} = -d_O\mathcal{G}(\boldsymbol{u}_O) - d_R\mathcal{G}(\boldsymbol{u}_R) \tag{6.35}$$

where the missing arguments (and the substitution of H with unity) are assumed. The other equation depends on the boundary conditions and these will now be gone through. They are controlled current, controlled potential with quasireversible and reversible reactions.

With controlled current, the value of $d_O\, G_O$ is controlled (*not* G_O itself!). Let the (dimensionless) value of this current be G. This yields the second equation simply as

$$G = d_O\mathcal{G}(\boldsymbol{C}'_O, n, H) = d_O\left(\mathcal{G}(\boldsymbol{u}_O, n, H) + C'_{O,0}d_O\mathcal{G}(\boldsymbol{v}_O, n, H)\right) \tag{6.36}$$

or, multiplying both sides by H and rearranging (using the briefer form)

$$d_OC'_{O,0}\mathcal{G}(\boldsymbol{v}_O) = GH - d_O\mathcal{G}(\boldsymbol{u}_O) \tag{6.37}$$

which can be used directly to obtain C'_O and thus, from (6.35), C'_R. Nevertheless for consistency with the other cases to follow, the equation system is given here:

$$\begin{bmatrix} d_O\mathcal{G}(\boldsymbol{v}_O) & d_R\mathcal{G}(\boldsymbol{v}_R) \\ d_O\mathcal{G}(\boldsymbol{v}_O) & 0 \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} -d_O\mathcal{G}(\boldsymbol{u}_O) - d_O\mathcal{G}(\boldsymbol{u}_R) \\ GH - d_O\mathcal{G}(\boldsymbol{u}_O) \end{bmatrix} . \tag{6.38}$$

Controlled potential can be either a potential step or some potential program such as LSV/CV, staircase voltammetry or even ac voltammetry, see the standard texts [74,257] for details. In all of these, we have (dimensionless) potential p at time T and the new potential p' at the next time level; p' might thus be a constant (as in potential step) or varying with time. One can then distinguish between the cases quasireversible (including irreversible) systems or fully reversible ones. Some simulation packages such as DigiSim [482] do not include the reversible case, arguing that it does not exist, and is in fact a quasireversible reaction with a large heterogeneous rate constant. This makes some sense but on the other hand, setting that rate at some some arbitrarily very high value to ensure reversible behaviour is no more justifiable than assuming Nernstian equilibrium, that is, reversibility.

A quasireversible system is characterised by the Butler-Volmer equation, here in dimensionless form,

$$d_OG' = K_fC'_{O,0} - K_bC'_{R,0} \tag{6.39}$$

with

$$K_f = K_0\exp(-\alpha p'), \qquad K_b = K_0\exp([1-\alpha]p') . \tag{6.40}$$

(Note that in (6.39), d_OG' is once again used, taking into account the diffusion coefficient of species O being different from that of the reference species).

Equation (6.39) is expanded for G' as before, the **u-v** substitutions made and rearranged, to give

$$(K_f - d_O\mathcal{G}(\boldsymbol{v}_O, n, H))\, C'_{O,0} - K_b C'_{R,0} = d_O\mathcal{G}(\boldsymbol{u}_O, n, H) \tag{6.41}$$

which, multiplying by H and adopting the previous shorthand for $\mathcal{G}$, and adding the flux condition (6.35), results in the system

$$\begin{bmatrix} -d_O\mathcal{G}(\boldsymbol{v}_O) & -d_R\mathcal{G}(\boldsymbol{v}_R) \\ HK_f - d_O\mathcal{G}(\boldsymbol{v}_O) & -HK_b \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} d_O\left(\mathcal{G}(\boldsymbol{u}_O) + \mathcal{G}(\boldsymbol{u}_R)\right) \\ d_O\mathcal{G}(\boldsymbol{u}_O) \end{bmatrix} . \tag{6.42}$$

This can easily be made into the irreversible case by setting $K_b = 0$ and, in principle, into the reversible case by setting K_0 very large. However, another way to ensure reversibility is to specify it as such, by the Nernst equation as in Chap. 2, page 14,

$$C'_{O,0}/C'_{R,0} = \exp(p') \tag{6.43}$$

or

$$C'_{O,0} - \exp(p')C'_{R,0} = 0 \tag{6.44}$$

which, added to the flux condition produces the system

$$\begin{bmatrix} d_O\mathcal{G}(\boldsymbol{v}_O) & d_R\mathcal{G}(\boldsymbol{v}_R) \\ 1 & -\exp(p') \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} -d_O\left(\mathcal{G}(\boldsymbol{u}_O) + \mathcal{G}(\boldsymbol{u}_R)\right) \\ 0 \end{bmatrix} . \tag{6.45}$$

6.3.1 Two-Point Derivative Cases

For those who prefer to keep the derivative approximation of G down to the two-point form, the above can perhaps be simplified a little; the **u-v** device is not needed as such, as only the first substitution (6.6) is required.

The flux condition (6.32) is represented in two-point form as

$$\frac{d_O}{H}\left(C'_{O,1} - C'_{O,0}\right) + \frac{d_R}{H}\left(C'_{R,1} - C'_{R,0}\right) = 0 \tag{6.46}$$

and (6.6) applied to the first two equations of (6.28), the C' at $X = H$ can be eliminated, to give, after some cleaning up, the flux condition equation

$$d_O\left(1 + \frac{1}{a'_{O,1}}\right) C'_{O,0} + d_R\left(1 + \frac{1}{a'_{R,1}}\right) C'_{R,0} = d_O\frac{b'_{O,1}}{a'_{O,1}} + d_R\frac{b'_{R,1}}{a'_{R,1}} \tag{6.47}$$

which is the one always needed out of the two. For controlled current, (6.36) becomes (again invoking (6.6))

$$GH = -\left(1 + \frac{1}{a'_{O,1}}\right) C'_{O,0} + \frac{b'_{O,1}}{a'_{O,1}} \tag{6.48}$$

or

$$\left(1+\frac{1}{a'_{O,1}}\right)C'_{O,0} = -GH + \frac{b'_{O,1}}{a'_{O,1}} \tag{6.49}$$

and thus, together with the first (6.47), this results in the system

$$\begin{bmatrix} d_O\left(1+\frac{1}{a'_{O,1}}\right) & d_R\left(1+\frac{1}{a'_{R,1}}\right) \\ \left(1+\frac{1}{a'_{O,1}}\right) & 0 \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} d_O\frac{b'_{O,1}}{a'_{O,1}} + d_R\frac{b'_{R,1}}{a'_{R,1}} \\ -GH + \frac{b'_{O,1}}{a'_{O,1}} \end{bmatrix} . \tag{6.50}$$

It is arguable whether this is in fact simpler than the form for general n, (6.38), but it does avoid calling a function.

The quasireversible case, analogous to (6.39), (6.40) and (6.41), becomes

$$\left[HK_f + d_O\left(1+\frac{1}{a'_{O,1}}\right)\right]C'_{O,0} - HK_bC'_{R,0} = d_O\frac{b'_{O,1}}{a'_{O,1}} \tag{6.51}$$

and the system to be solved,

$$\begin{bmatrix} d_O\left(1+\frac{1}{a'_{O,1}}\right) & d_R\left(1+\frac{1}{a'_{R,1}}\right) \\ HK_f + d_O\left(1+\frac{1}{a'_{O,1}}\right) & -HK_b \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} d_O\frac{b'_{O,1}}{a'_{O,1}} + d_R\frac{b'_{R,1}}{a'_{R,1}} \\ d_O\frac{b'_{O,1}}{a'_{O,1}} \end{bmatrix} . \tag{6.52}$$

Again, the irreversible case is accommodated by setting K_b to zero.

The reversible case gives rise to the same equation as for higher n as in (6.44) and thus to the system

$$\begin{bmatrix} d_O\left(1+\frac{1}{a'_{O,1}}\right) & d_R\left(1+\frac{1}{a'_{R,1}}\right) \\ 1 & -\exp(p') \end{bmatrix} \begin{bmatrix} C'_{O,0} \\ C'_{R,0} \end{bmatrix} = \begin{bmatrix} d_O\frac{b'_{O,1}}{a'_{O,1}} + d_R\frac{b'_{R,1}}{a'_{R,1}} \\ 0 \end{bmatrix} . \tag{6.53}$$

6.4 Two Species with Coupled Reactions. U-V

Up to this point, the treatments have involved reactions for which the discrete form of the reaction-diffusion equations involve only terms in concentration of the species to which the discrete equation applies. That is, if there were two substances involved, O and R as above, then the discrete equation at a point i had terms only in $C'_{O,.}$ for species O, and only $C'_{R,.}$ for species R. This made it possible to use the Thomas algorithm to reduce a system like (6.27) to (6.28), treating the two species' systems separately. They then get coupled through the boundary conditions.

When homogeneous reactions take place, it often happens that some of the discrete equations contain terms in concentration for more than the one species, and it is then not generally possible to use the Thomas algorithm to reduce the systems. These systems are said to be coupled. An example will illustrate this situation.

Consider the catalytic or EC′ reaction pair as described in Sect. 2.4, page 23, (2.76). For generality, the species designations A and B are now written as O and R, and the reaction pair then is

$$\begin{aligned} O + e^- &\Leftrightarrow R \\ R &\rightarrow O \ . \end{aligned} \tag{6.54}$$

The derivation of the discrete equations corresponding to this reaction pair will be given in Chap. 8 and it will suffice here to provide the general form they will take:

$$\begin{aligned} C'_{O,i-1} + a_{O,1}(i)C'_{O,i} + a_k(i)C'_{R,i} + a_{O,2}(i)C'_{O,i+1} &= b_{O,i} \\ C'_{R,i-1} \quad + (a_{R,1}(i) - a_k(i))C'_{R,i} + a_{R,2}(i)C'_{R,i+1} &= b_{R,i} \ . \end{aligned} \tag{6.55}$$

The coefficients $a_{.,1}(i)$ and $a_{.,2}(i)$ arise from the particular spatial approximation of the second derivative, while the $a_k(i)$ come from the homogeneous chemical reaction rate, as will be described in Chap. 8.

It will always be the case that the extra term, as seen here in the first equation for species O, lies at index i only. As stated above, it is not generally possible to start at the outer limit for X and reduce these two equations to fewer unknowns, as with uncoupled cases. In fact, in this particular case, this can be done, using a slightly complicated trick but this will not be dealt with here.

There are the usual boundary conditions depending on the experiment performed on this system. One possible way to handle all this is simply to write out the whole system as a large linear system, expand that to include the boundary conditions, and solve. This, "brute force" approach (see below), has in fact been used [138] and can even be reasonably efficient if the number of equations is kept low, by use, for example, of unequal intervals, described in Chap. 7. If the equations in such a system are arranged in the order as above (6.55), it will be found that it is tightly banded, except for the first two rows for the boundary conditions, which may have a number of entries up to the number n used for the current approximation.

A better alternative approach is what will be called the **Rudolph** method [476], after the person who introduced it into electrochemical simulation. It was known before 1991 under various names, notably block-tridiagonal [280, 412, 470, 471, 528, 570]. This comes from the fact that if one lumps the large matrix into a matrix of smaller matrices and vectors, the result is a tridiagonal system that is amenable to more efficient methods of solution. In the present context, we define some vectors

$$\boldsymbol{C}'_x \equiv \begin{bmatrix} C'_{O,x} \\ C'_{R,x} \end{bmatrix} \tag{6.56}$$

where x can be $i-1$, i or $i+1$. The equation pair (6.55) can be partitioned into three vertical slices involving such vectors, and then rewritten in the matrix-vector form,

$$\boldsymbol{C'}_{i-1} + \boldsymbol{A}_i\boldsymbol{C'}_i + \boldsymbol{a}_2\boldsymbol{C'}_{i+1} = \boldsymbol{B}_i \tag{6.57}$$

in which we have another vector and two matrices for the coefficients:

$$\boldsymbol{B}_i \equiv \begin{bmatrix} b_{O,i} \\ b_{R,i} \end{bmatrix} , \tag{6.58}$$

$$\boldsymbol{A}_i \equiv \begin{bmatrix} a_{O,1}(i) & a_k(i) \\ 0 & a_{R,1}(i) - a_k(i) \end{bmatrix} , \tag{6.59}$$

and

$$\boldsymbol{a}_2(i) \equiv \begin{bmatrix} a_{O,2}(i) & 0 \\ 0 & a_{R,2}(i) \end{bmatrix} . \tag{6.60}$$

The point of this exercise is that (6.57) now is very like the previous single-species (6.2), except that it involves concentration vectors and coefficient vectors and matrices, rather than all scalars as in (6.2). In Chap. 8, details will be given on how this equation can be reduced to two terms; suffice it to say here that the process is analogous to that for a single species, making use of the Nth equation containing the outer boundary vector $\boldsymbol{C}_{N+1}$. The result is that the system (6.57) is replaced by a new reduced system, the first few equations of which are

$$\begin{aligned} \boldsymbol{C'}_0 + \boldsymbol{A'}_1\boldsymbol{C'}_1 &= \boldsymbol{B'}_1 \\ \boldsymbol{C'}_1 + \boldsymbol{A'}_2\boldsymbol{C'}_2 &= \boldsymbol{B'}_2 \\ \boldsymbol{C'}_2 + \boldsymbol{A'}_3\boldsymbol{C'}_3 &= \boldsymbol{B'}_3 \\ &\cdots \end{aligned} , \tag{6.61}$$

very similar in form to the system (6.3). Not surprisingly, the **u-v** device used for the single-species case can be devised for the matrix-vector case, and will be called the **U-V** device. It turns out that $\boldsymbol{U}$ becomes a vector and $\boldsymbol{V}$ a matrix. We start by defining starting values:

$$\boldsymbol{U}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.62}$$

and

$$\boldsymbol{V}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.63}$$

which clearly allows the tautological statement analogous to (6.15),

$$\boldsymbol{C'}_0 = \boldsymbol{U}_0 + \boldsymbol{V}_0\boldsymbol{C'}_0 . \tag{6.64}$$

Looking at (6.62), first line, we can, analogously to (6.6), write the vector $\boldsymbol{C'}_1$ as a linear function of $\boldsymbol{C'}_0$, that is,

$$\boldsymbol{C'}_1 = \boldsymbol{A'}^{-1}\boldsymbol{B'}_1 - \boldsymbol{A'}^{-1}\boldsymbol{C'}_0 \tag{6.65}$$

which is then rewritten as

$$\boldsymbol{C'}_1 = \boldsymbol{U}_1 + \boldsymbol{V}_1 \boldsymbol{C'}_0 \tag{6.66}$$

with obvious definitions for the $\boldsymbol{U}_1$ and $\boldsymbol{V}_1$, in view of (6.65) and (6.64). This process can be repeated for indices i equal to $2, 3, \ldots$, and the general recursive formulae for the $\boldsymbol{U}$ and $\boldsymbol{V}$ are as follows:

$$\boldsymbol{U}_i = \boldsymbol{A'}^{-1} \left(\boldsymbol{B}'_i - \boldsymbol{U}_{i-1} \right) \tag{6.67}$$

and

$$\boldsymbol{V}_i = -\boldsymbol{A'}^{-1} \boldsymbol{V}_{i-1} \, . \tag{6.68}$$

The process (which, as will be seen and already seen in the uncoupled case of the **u-v** device above) needs to be carried forward only to $i = n - 1$. It yields $n - 1$ equations

$$\boldsymbol{C}'_i = \boldsymbol{U}_i + \boldsymbol{V}_i \boldsymbol{C}'_0 \tag{6.69}$$

which can finally be used for the boundary conditions.

We are still dealing with two species as in the uncoupled case and the same boundary conditions apply; they are reformulated in the present matrix-vector form here. As noted above, there is a common condition for all experiments, the flux condition (6.30), generalised to include the normalised diffusion coefficients, to the gradient condition (6.32), and we now write out its discrete form fully, pairing the two species' terms for each spatial index:

$$d_O \beta_0 C'_{O,0} + d_R \beta_0 C'_{R,0} + \cdots + d_O \beta_{n-1} C'_{O,n-1} + d_R \beta_{n-1} C'_{R,n-1} = 0 \tag{6.70}$$

and invoking vector notation, writing $\boldsymbol{C}'_i$ for $[C'_{O,i} \; C'_{R,i}]^T$, this becomes

$$\beta_0 [d_O \;\; d_R] \boldsymbol{C}'_0 + \beta_1 [d_O \;\; d_R] \boldsymbol{C}'_1 + \cdots + \beta_{n-1} [d_O \; d_R] \boldsymbol{C}'_{n-1} = 0 \tag{6.71}$$

which will be one of the two equations needed for all cases. It will be combined with the particular equations for the cases Cottrell, quasi/irreversible and controlled current.

For the simple Cottrell case, we have

$$C'_{O,0} = 0 \tag{6.72}$$

and, for convenience in what follows, this is expanded to include the other species,

$$C'_{O,0} + \left(0 \cdot C'_{R,0} \right) = 0 \tag{6.73}$$

or

$$\beta_0 [1 \;\; 0] \boldsymbol{C}'_0 = 0 \tag{6.74}$$

in vector form (multiplying by β_0 for convenience). At the risk of repetition but in order to make the next step clear, this equation is now paired with (6.71):

$$\beta_0[1\ \ 0]\boldsymbol{C}'_0 = 0$$
$$\beta_0[d_O\ \ d_R]\boldsymbol{C}'_0\ \ +\beta_1[d_O\ \ d_R]\boldsymbol{C}'_1 + \ldots\ +\beta_{n-1}[d_O\ \ d_R]\boldsymbol{C}'_{n-1} = 0$$

and these can be combined in one vector-matrix equation,

$$\beta_0 \begin{bmatrix} 1 & 0 \\ d_O & d_R \end{bmatrix} \boldsymbol{C}'_0 + \beta_1 \begin{bmatrix} 0 & 0 \\ d_O & d_R \end{bmatrix} \boldsymbol{C}'_1 + \cdots + \beta_{n-1} \begin{bmatrix} 0 & 0 \\ d_O & d_R \end{bmatrix} \boldsymbol{C}'_{n-1} = 0\,. \quad (6.75)$$

This is now written in the more general form,

$$\beta_0 \boldsymbol{M}_0 \boldsymbol{C}'_0 + \beta_1 \boldsymbol{M}_1 \boldsymbol{C}'_1 + \cdots + \beta_{n-1} \boldsymbol{M}_{n-1} \boldsymbol{C}'_{n-1} = 0 \quad (6.76)$$

with

$$\boldsymbol{M}_0 = \beta_0 \begin{bmatrix} 1 & 0 \\ d_O & d_R \end{bmatrix} \quad (6.77)$$

and

$$\boldsymbol{M}_i = \beta_i \begin{bmatrix} 0 & 0 \\ d_O & d_R \end{bmatrix} \quad (6.78)$$

for all $0 < i < n$. This equation contains the vectors $\boldsymbol{C}'_i$, and we can now apply the **U-V** relations (6.69) to put it all in terms of the one unknown vector $\boldsymbol{C}'_0$,

$$\beta_0 \boldsymbol{M}_0 \left(\boldsymbol{U}_0 + \boldsymbol{V}_0 \boldsymbol{C}'_0\right) + \cdots + \beta_{n-1} \boldsymbol{M}_{n-1} \left(\boldsymbol{U}_{n-1} + \boldsymbol{V}_{n-1} \boldsymbol{C}'_0\right) = 0 \quad (6.79)$$

and, defining the matrices

$$\boldsymbol{P} \equiv \sum_{i=0}^{n-1} \beta_i \boldsymbol{M}_i \boldsymbol{V}_i\,, \quad (6.80)$$

$$\boldsymbol{Q} \equiv -\sum_{i=0}^{n-1} \beta_i \boldsymbol{M}_i \boldsymbol{U}_i\,, \quad (6.81)$$

(6.79) then becomes

$$\boldsymbol{P}\boldsymbol{C}'_0 = \boldsymbol{Q} \quad (6.82)$$

which can readily be solved for the boundary values.

It will be seen that the equation always takes this form except for constant current and in that case, only a slightly different one. The differences lie in the definitions of the $\boldsymbol{M}$ matrices.

For the reversible case, apart from the flux condition, there is the Nernst equation, previously shown to be

$$C'_{O,0} - \exp(p')C'_{R,0} = 0 \quad (6.83)$$

now written as

$$[1\quad -\exp(p')]\,\boldsymbol{C}'_0 = 0 \quad (6.84)$$

which is combined with the flux (6.71). This time the two equations are not presented, because they follow the above pattern for the Cottrell case, ending with the same (6.82), with $\boldsymbol{P}$ and $\boldsymbol{Q}$ generated as sums as in (6.80) and (6.81), the difference being in the first $\boldsymbol{M}$, here given by

$$\boldsymbol{M}_0 = \beta_0 \begin{bmatrix} 1 & \exp(p') \\ d_O & d_R \end{bmatrix} \tag{6.85}$$

and the other $\boldsymbol{M}_i$ exactly as in (6.78).

For the quasireversible case, the flux condition is combined with the Butler-Volmer equation, as given above in (6.39), (6.40) and (6.41), the latter now to be written in long-hand as

$$\beta_0 C'_{O,0} + \beta_1 C'_{O,1} + \cdots + \beta_{n-1} C'_{O,n-1} = \frac{K_f H}{d_0} C'_{O,0} - \frac{K_b H}{d_R} C'_{R,0} \tag{6.86}$$

and collecting terms, again pairing the two boundary values,

$$\beta_0 \left[\left(1 - \frac{K_f H}{d_O \beta_0} \right) C'_{O,0} + \frac{K_b H}{d_0 \beta_0} C'_{R,0} \right] + \beta_1 C'_{O,1} + \cdots + \beta_{n-1} C'_{O,n-1} = 0 \tag{6.87}$$

which leads again to the same equations, with the only difference here lying in $\boldsymbol{M}_0$, now given by

$$\boldsymbol{M}_0 = \begin{bmatrix} \left(1 - \frac{K_f H}{d_O \beta_0} \right) & \frac{K_b H}{d_O \beta_0} \\ d_O & d_R \end{bmatrix} \tag{6.88}$$

and again, the other $\boldsymbol{M}_i$ as in (6.78).

The totally irreversible case is again obtained by setting K_b to zero in the above equations.

This leaves the controlled current case. As noted above (Sect. 6.3), it is the current, not the gradient, that is controlled, so the equation is

$$G = \frac{d_0}{H} \sum_{i=0}^{n-1} \beta_i C'_{O,i} \tag{6.89}$$

where G is the dimensionless current imposed. So, expanding the sum, working in the (zero-weighted) terms in $C'_{R,i}$ and going straight into the vector notation, this becomes

$$\beta_0 \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{C}'_0 + \beta_1 \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{C}'_1 \cdots + \beta_{n-1} \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{C}'_{n-1} = GH/d_0 \tag{6.90}$$

and we note that this differs from all the equations up til now in this section, in that the right-hand side is not zero. Combined with the inevitable flux condition (6.71), this yields the slightly different matrix/vector equation

$$\boldsymbol{P}\boldsymbol{C}'_0 = \begin{bmatrix} \frac{GH}{d_0} \\ 0 \end{bmatrix} + \boldsymbol{Q} \tag{6.91}$$

again readily solved.

6.5 Brute Force

In the last two sections, we have applied increasingly tricky devices to solve what amounts to systems of equations, in order to make their solution efficient. Even the two-species uncoupled case involved the generation of the $\boldsymbol{u}$ and $\boldsymbol{v}$ vectors and the solution of small (2×2) systems to obtain the boundary values. In the case of coupled systems, the problems mount and matrix-vector equations had to be used.

One might ask, is all this necessary? It is not. In the next chapter, unequal intervals are described, and these make it possible to reduce the number of points N in space to quite reasonably small values. Whereas with equal intervals we might need some hundreds of sample points along X, with unequal intervals we can make do with as few as 15 or so. This means that systems such as (6.2) have only as few equations as that or, with two species involved, twice that many (not counting the boundary equations for the moment). The full systems are strongly banded if the equations are ordered suitably, and this banding invites the use of more efficient methods of solution, but if computers can do it fast enough by simply solving the whole system without regard to the bandedness, tricks might not be needed. All we then need to do is to add the equations for the boundary conditions (that is, include the C'_0 variables in the system) and call a matrix solver for the solution. This was mentioned by Rudolph [477] but at the time considered "prohibitively expensive" in computer time. Meanwhile, however, computers have become much faster and the idea has been investigated [138] and found, in some cases, to be about as good as the tricky methods and in some cases even better. As well, there are some reasonably simple methods of varying complexity to make such whole-system solutions more efficient. These will be mentioned below. First, a more concrete description of what is meant is presented here.

A simple example would be the single-species case and Cottrell. Then, the system (6.2) would be augmented by adding, at the top, the Cottrell condition $C'_0 = 0$ and the matrix equation is

$$\begin{bmatrix} 1 & 0 & \dots & & & \\ 1 & a_{1(1)} & a_{2(1)} & 0 & \dots & \\ 0 & 1 & a_{1(2)} & a_{2(2)} & 0 & \dots \\ & & \ddots & & \ddots & \\ & & & 1 & a_{1(N-1)} & a_{2(N-1)} \\ & & & & 1 & a_{1(N)} \end{bmatrix} \begin{bmatrix} C'_0 \\ C'_1 \\ C'_2 \\ \vdots \\ C'_{N-1} \\ C'_N \end{bmatrix} = \begin{bmatrix} 0 \\ b_1 \\ b_2 \\ \vdots \\ b_{N-1} \\ b_N - a_{2(N)}C'_{N+1} \end{bmatrix} \tag{6.92}$$

where the known outer boundary term, C'_{N+1} has been carried over to the vector of knowns. This is a trivial example but serves to explain the idea. It is in fact a tridiagonal system, amenable to the Thomas algorithm to be described in Chap. 8. Extension to the other more complicated cases is obvious, except where there are two species, when the optimal order of the unknowns vector $\boldsymbol{C'}$ is to pair the terms for the two species, that is,

$$\boldsymbol{C}' \equiv \left[C'_{O,0}\, C'_{R,0}\, C'_{O,1}\, C'_{R,1} \ldots\, C'_{O,N}\, C'_{R,N}\right]^T \tag{6.93}$$

which ensures tight banding.

Extension to the other less trivial cases appears straightforward. Most boundary conditions will put up to $2n$ elements into the first and second rows. For those cases involving coupled equations, the rows after the second will contain five elements. It can be seen from the first row of (6.55), that there needs to be a zero inserted after the $C'_{O,i-1}$, for the nonexistent $C'_{R,i-1}$, but not a similar insertion after the last element, or a total of five. Thus, if one eliminates the excess elements in the first two rows, one can then use a solver for pentadiagonal systems, which is also quite feasible.

6.6 A General Formalism

Sometimes, when trying out a new method when efficiency is not (initially) of highest priority, or when doing a stability study, it can be of advantage to have a general formula for all possible boundary conditions. An early use of such a formula is seen in [472], and the formula is also seen in some texts such as [528]. In the electrochemical context, it has been presented a few times in recent years [116, 152, 529]. The formula is given in the form of [116]

$$g + rc_0 - d\left(\frac{\partial c(0,t)}{\partial x}\right) = 0\,. \tag{6.94}$$

The constants g, r and d can take on various values to express any given boundary condition. Thus, if we set $g = d = 0$, we are left with $c_0 = 0$, the Dirichlet (Cottrell) condition; if we set $r = 0$ and $d = 1$, we have the Neumann or controlled current condition; and setting $g = 0$ gives us Robin conditions. The constant r expresses the heterogeneous rate constant (this formula only considers a single species, so an irreversible reaction is implied).

The Cottrell case is simple, and needs no further comment. The other two cases can be usefully expressed in a different manner. The derivative is expressed as the n-point approximation, giving

$$g + rc_0 - d\sum_{i=0}^{n-1} \beta_i c_i = 0 \tag{6.95}$$

or, removing the c_0 element from the sum,

$$g + (r - d\beta_0)\, c_0 - d\sum_{i=1}^{n-1} \beta_i c_i = 0 \tag{6.96}$$

giving

$$c_0 = \frac{d\sum_{i=1}^{n-1} \beta_i c_i \quad - g}{r - d\beta_0} \tag{6.97}$$

and dividing by $-\beta_0$ and setting $d = 1$,

$$c_0 = \frac{-b}{\beta_0} \left\{ \sum_{i=1}^{n-1} \beta_i c_i \quad - g \right\} \tag{6.98}$$

with

$$b = \left(1 - \frac{r}{\beta_0} \right)^{-1} . \tag{6.99}$$

The convenient thing here is that we now have the whole spectrum of conditions from a very fast reaction ($b = 0$, implying $r \to \infty$), through medium fast reactions (medium values of b and thus r) to the controlled current case ($b = 1$). The first case also encompasses the Cottrell case.

7 Unequal Intervals

In the preceding chapters, a grid with equal intervals in both time and space was assumed (Fig. 1.1, page 3). There are several reasons for deviating from equal space intervals. Firstly, one wants both to minimise the number of points over the concentration profile and, at the same time, to have close spacing near the electrode. Secondly, in some simulations – but not always – there arise sharp concentration changes somewhere in the diffusion space, usually adjacent to the electrode. One then wants to have close spacing in such regions in order to be able to simulate concentration changes at all. This points to adaptive techniques; but first the simpler fixed unequal grid techniques will be dealt with.

In this chapter, only one-dimensional unequal intervals will be described. Mapping techniques for two-dimensional simulations are left to Chap. 12.

Consider Fig. 2.4 on p.16, showing the concentration profile for a Cottrell simulation at different times. It is clear that especially the profiles at small T values are strongly compressed near the electrode, and that equal intervals in X would be wasteful at larger X. An unequal spacing of the intervals could not only provide more detail near the electrode where it is needed, but also make do with fewer points by wide spacing far away from the electrode. So some kind of grid stretching is indicated on this account.

If there are homogeneous chemical reactions, they may give rise to reaction layers that can, for high reaction rates, be very thin. In order to get reasonable simulation results, at least a few points are needed within that layer. If equal intervals in X are used, this means using a very large value of N and correspondingly long computation times. With unequal intervals, fewer points will do. One needs to have an idea of the thickness, μ^*, defined on page 24 (2.80), and set the position of the points accordingly, as described in the following sections.

There are several approaches to implementing grid stretching. The two competing approaches are (1) the direct application of a stretched grid, discretising directly on that unequal grid, and (2) the transformation of the equation to new coordinates and using equal intervals there. There is a wealth of literature on this subject. Noye [422], and Hunter and Jones [312] recommend transformation, as does an early study by Crowder and Dalton [186]. The much cited comparison by Kalnay de Rivas [328] reached the same

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 103–117 (2005)
www.springerlink.com

conclusion. However, Rudolph [478] showed conclusively that, under the conditions of electrochemical simulations at least, the situation is the reverse. He showed that both the current approximation as well as the second, spatial, derivative as computed directly from an unequally spaced grid, are more accurate than those computed from a transformed grid with equal intervals. The reason for the better performance of direct discretisation appears to be that concentration profiles tend to be close to linear near the electrode, so that the current approximation can be calculated quite well with only a few points (Rudolph always uses just two), whereas in the transformed space (see below, Sect. 7.1), the profile becomes curved near the electrode and more points are needed for a good approximation. Exactly why the second, spatial, derivative is also more accurate when calculated directly, is not clear. Numerical experiments performed by the present author show that, for several different (artificial) profile functions, including the realistic one of $\mathrm{erf}(x)$, which often resembles real concentration profiles, the second derivative calculated on the transformed grid is poor, especially near the electrode, where accurate values are most needed. Direct calculation on the unequal grid yields roughly the same accuracy right across the profile.

As well, it will be seen that the formulation of the second spatial derivative on a general grid, spaced in some unspecified way, is rather flexible and permits easy replacement, in a given program, of the stretching function used including, if one desires, equal spacing, or even arbitrary placement of each point.

7.1 Transformation

Transformation for electrochemical work was proposed in the now classic paper by Joslin and Pletcher [321]. They described a transformation, say from X to Y, such that equal intervals in Y are a mapping of (correspond to) unequal intervals in X. The aim is to find a transformation function which produces in Y-space a concentration profile that resembles a straight line as much as possible.

The general treatment is as described by Joslin and Pletcher [321]. Assume an arbitrary transformation function, $f(X)$ mapping points in X onto the new axis Y and its inverse, $g(Y)$. Then the right-hand diffusion term in the diffusion equation, $\frac{\partial^2 C}{\partial X^2}$ becomes, by the rules of elementary calculus,

$$\frac{\partial}{\partial X}\left(\frac{\partial C}{\partial X}\right) = \frac{1}{g'(Y)}\frac{\partial}{\partial Y}\left(\frac{1}{g'(Y)}\frac{\partial C}{\partial Y}\right) . \tag{7.1}$$

This can be expanded further to

$$\frac{\partial}{\partial X}\left(\frac{\partial C}{\partial X}\right) = \frac{1}{g'(Y)}\left(\frac{1}{g'(Y)}\frac{\partial^2 C}{\partial Y^2} + \left(\frac{\partial}{\partial Y}\left(\frac{1}{g'(Y)}\right)\right)\frac{\partial C}{\partial Y}\right) . \tag{7.2}$$

The work of Feldberg [231] indirectly provides a useful transformation function, that has some convenient properties. The function is

$$f(X) = Y = \ln(1 + aX) \tag{7.3}$$

where a is an adjustable parameter. Inserting this into (7.2), recognising that

$$g(Y) = (e^Y - 1)/a \qquad \text{and thus} \qquad g'(Y) = e^Y/a\ , \tag{7.4}$$

the new dimensionless diffusion equation in Y-space is then

$$\frac{\partial C}{\partial T} = a^2 e^{-2Y} \left(\frac{\partial^2 C}{\partial Y^2} - \frac{\partial C}{\partial Y} \right) . \tag{7.5}$$

Note that if the original equation to be solved contains homogeneous chemical terms, these do not change upon transforming the equation, since they give rise to additional terms not involving X or Y.

The transformation function (7.3) is mathematically (approximately) equivalent to the Feldberg stretching function (7.16), as is shown in Appendix B, where the relation between the respective adjustable parameters is given.

The gradient G is conveniently calculated on the grid in Y, and it is easy to show that this is simply

$$G = \left.\frac{\partial C}{\partial X}\right|_{X=0} = a \left.\frac{\partial C}{\partial Y}\right|_{Y=0} \tag{7.6}$$

which seems very convenient, requiring only a call to the routine that evaluates the $\mathcal{G}$ function in Y-space and a multiplication by the parameter a. The problem, as Rudolph showed [478], is that this yields a poor G-value, unless a large n is used (6 or even 7). This is not a bad thing in itself, as we have functions for G that we can simply call. However, derivative boundary conditions involving many points become messy. Rudolph uses just two points, arguing that if the first point is sufficiently close to the electrode, as it is with severe stretching, two points are good enough, and this simplifies the discretisation of the boundary conditions a lot. There are some arguments for using $n = 3$; Bieniasz [95,100] points out that if a second-order second spatial derivative is used for the simulation, then a matching second-order (3-point) G-approximation is best. On the other hand, the second spatial derivative directly discretised on an unequal grid is in fact a first-order approximation, arguing for Rudolph's two-point G. This will be a matter of individual choice.

7.1.1 Discretising the Transformed Equation

Transformation (7.3) leads to the new diffusion (7.5) in Y-space. Although it is fairly obvious how the new right-hand side is discretised, for completeness, this will be described here.

Instead of a number of sample points in X, we now have a number of equally spaced points along the new coordinate Y with a spacing of δY. Without considering which simulation algorithm is to be used, we discretise the new (7.5) at the point Y_i as follows:

$$\delta C_i \approx \delta T a^2 \exp(-2Y_i) \left(\frac{C_{i-1} - 2C_i + C_{i+1}}{\delta Y^2} - \frac{C_{i+1} - C_{i-1}}{2\delta Y} \right) \tag{7.7}$$

and given that $Y_i = i\,\delta Y$, this rearranges to

$$\delta C_i \approx \lambda_i \Big(\big(1 + \tfrac{1}{2}\delta Y\big) C_{i-1} - 2C_i + \big(1 - \tfrac{1}{2}\delta Y\big) C_{i+1} \Big) \tag{7.8}$$

with λ_i defined as

$$\lambda_i = a^2 \exp(-2i\delta Y) \frac{\delta T}{\delta Y^2} \,. \tag{7.9}$$

The coefficients in the right-hand term in brackets in (7.8) can be precomputed, as can the row of λ_i values. Further details of how all this is implemented are given in Chap. 8 for the respective simulation algorithms.

As mentioned above, Rudolph [478] pointed out that this discretisation yields very poor values and ultimately to poor simulation performance, compared to direct discretisation on an uneven grid, see below. Tests show that particularly at small X values, near the electrode where the greatest changes occur, the second spatial derivatives as seen in (7.7) are approximated very poorly. Rudolph [479,480] and Bieniasz [107] showed that if what we might call the semi-transformed (7.1) is used, rather than the fully transformed equation, this problem is eliminated. Doing this in a consistent manner, and assuming general transformation functions $f(X)$ and $g(Y)$, we can write for the ith point the approximation

$$\frac{1}{g'(Y)} \frac{\partial}{\partial Y} \left(\frac{1}{g'(Y)} \frac{\partial C}{\partial Y} \right) \approx \frac{1}{g'(Y)} \frac{1}{\delta Y} \left(\frac{C_{i+1} - C_i}{g'\Big(Y_{i+\frac{1}{2}}\Big)\delta Y} - \frac{C_i - C_{i-1}}{g'\Big(Y_{i-\frac{1}{2}}\Big)\delta Y} \right) . \tag{7.10}$$

Using the transformation (7.3) and thus substituting for $g'(Y)$ as given in (7.4) at the indices given and rearranging a little, this becomes

$$\delta C_i \approx \lambda_i \Big(\exp\big(-\tfrac{1}{2}\delta Y\big)(C_{i+1} - C_i) - \exp\big(\tfrac{1}{2}\delta Y\big)(C_i - C_{i-1}) \Big) \tag{7.11}$$

with λ_i as defined above (7.9). Some tests indicate that this is a much better approximation, giving derivatives of roughly the same accuracy over the whole X-range. The accuracy is comparable to that of direct discretisation on the uneven grid, described below. Incorporation into the whole diffusion equation, as was done for the completely transformed diffusion equation in (7.8), is obvious from here on.

7.1.2 The Choice of Parameters

We have seen from the above that, in some way or other, we choose the value of $H_1 = X_1$. We also have a maximum value X_L along X, which depends on the experiment. Using (7.3), these two values provide the two equivalent values in Y-space. The Y-value corresponding to X_1 is also the interval in Y, as these are all equal. The equations are

$$\delta Y = \ln(1 + aX_1) \tag{7.12}$$

and

$$Y_N = \ln(1 + aX_L) \tag{7.13}$$

which set the number of intervals in Y,

$$N = Y_N/\delta Y \tag{7.14}$$

(rounded up, thus correcting Y_N slightly). Knowing X_1, there are then two parameters to be determined, a and N. These are dependent on each other, so the choice of one sets the other.

The easy alternative is to set a. One develops a feeling for what value might be a good one. Having set this value and knowing that of X_L, the above (7.12) ... (7.14) yield N.

Alternatively, one might want to set X_1 and N and find an a value that provides these. This is a little harder. Substituting for Y_N from (7.14) in (7.12) and combining (7.13) with the result, we obtain the function

$$f(a) = N \ln(1 + aX_1) - \ln(1 + aX_L) \tag{7.15}$$

which can be solved for that a which gives $f(a) = 0$. This needs to be done numerically. There are two solutions. The trivial (and unwanted) solution is $a = 0$. What makes the calculation rather easy is the fact that we do not need a very accurate value for the a parameter. So a rough binary search will very quickly find a suitable value (see such elementary texts on numerical computing as [163, 266, 452]). A binary search will be found better here than the generally more efficient Newton method, which can point in the wrong direction and converge on the trivial solution, or lead to numerical problems (negative arguments to the log function).

Lastly, it is possible also to set a, X_L and N, and to use them to find δY and thereby X_1. If it is done on a calculator beforehand, one sees what value results, before committing the chosen parameters to a simulation run.

7.2 Direct Application of an Arbitrary Grid

A stretched stack of boxes was used by Feldberg [231] for the box-method, to be described in Chap. 9. Pao and Dougherty [433] developed the same idea

(and stretching function) in 1969, in the context of fluid dynamic simulations. This is the simple placement of points at increasing intervals, in some suitable point distribution or stretching function, and discretisation of the second derivative of concentration along X on that unequal grid.

There are various ways of specifying the stretched point placement. The current favourite appears to be the exponentially expanding sequence of Feldberg [232], in which there essentially is a sequence of intervals H along X,

$$H_i = H_{i-1}\,\gamma \tag{7.16}$$

or

$$H_i = H_1 \gamma^{i-1} \tag{7.17}$$

starting at some chosen H_1 and choosing the stretching parameter γ suitably. In Feldberg's case, the points thus generated are in fact box walls, but one could equally well use them with the point method as concentration nodes. Also, Feldberg uses a slightly different notation, setting not the γ used here, but the related parameter $\beta = \ln(\gamma)$. The value of γ is chosen such that H_1 is rather small, but the number of points in the diffusion region is also rather small. While with equal intervals, some hundreds of points might be needed, a suitable choice of γ (for example, 1.2–1.5) can reduce their number to 10–20 (Feldberg suggests a β range $0 \ldots 0.5$).

As will be seen below, the way stretched intervals are used here is that a set of positions in X are specified. We must therefore convert the intervals formula above (7.17) to one in terms of X. For any $N > 0$,

$$X_N = H_1 \sum_{k=1}^{N} \gamma^{k-1} = H_1 \sum_{k=0}^{N-1} \gamma^{k} \tag{7.18}$$

and this is readily summed to give the expression

$$X_N = H_1 \frac{\gamma^N - 1}{\gamma - 1} \tag{7.19}$$

which is also the expression seen in Feldberg [231], albeit in terms of β.

The drawback of this point sequence (and most others except a sequence of equal intervals) is that the approximation to the second derivative with respect to X is then a first-order approximation, as was mentioned in Chap. 3, Sect. 3.8. The use of more than three points is thus indicated, and such approximations are described in Chaps. 3 and 9, and some formulas are given in Appendix A.

There is one unequal sequence of points for which the second derivative, when applied directly to the points, retains the second-order nature of an even point spacing. This was found by Sundqvist and Veronis [537] in 1970. Their stretching function was

$$H_i = H_{i-1}(1 + \alpha H_{i-1})\ . \tag{7.20}$$

In the original form, the α factor was effectively normalised by dividing by the total extension X_L of the diffusion space. In the present context, a suitable normalisation might be division by H_1, giving

$$H_i = H_{i-1}(1 + \alpha H_{i-1}/H_1) \; . \tag{7.21}$$

Then, this function will yield sequences somewhat similar to exponentially expanding sequences, by taking α equal to something like $\frac{1}{2}(\gamma - 1)$. This sequence has not become popular (perhaps because it has escaped notice). It might, however, be a useful alternative.

Interestingly, Saul'yev mentions [496, p. 149] a private communication from A. A. Samarskii, who found precisely the same relation (7.20) and that it permits second-order approximations to the three-point second derivative.

A comparison of the two functions discussed here is shown in Fig. 7.1, presenting the distribution out to about $X = 6$ for the exponential sequence (7.16) for $\gamma = 1.5$, and the S&V sequence (7.21) for $\alpha = 0.2$. Both were started with a first interval H_1 of 0.05. The exponential sequence gives 10 points, ending at $X = 5.67$ and the other sequence gives 9 points ending at $X = 6.36$. It is seen that the S&V sequence gives a more drastic unevenness in the spacing. Preliminary numerical experiments by the present author indicate that the second spatial derivatives on the S&V sequence are indeed mostly very accurate, but decline in accuracy at large X. In two comparison programs, one using the exponentially expanding and the other the S&V sequence, both starting with a base interval of 0.01 and using 50 points in the X range 0-6 and discretising directly, a Cottrell simulation using 100 steps in time each of length 0.01, the exponentially expanding sequence showed an error in the final current at $T = 1$ of about 10^{-3} (relative), whereas the S&V sequence's error was 10^{-2}. So, it appears that this points sequence might not be so good.

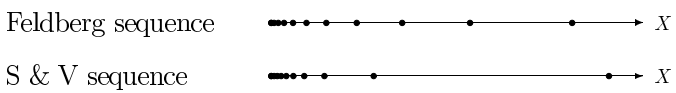


Fig. 7.1. Points spaced unequally with the two functions

There is an inherently stretched grid implementation in the simulation technique called orthogonal collocation, to be discussed in Chap. 9. It will be seen that this can be extremely efficient but it suffers, as all fixed stretched grids do, from inflexibility, as is noted in general in Sect. 7.3.

An interesting special case, mentioned in Chap. 3, is that of the second derivative on four points, $u_2''(4)$. For arbitrarily (unequally) spaced points, this is a second-order accurate approximation and, as described in Chap. 9, it has some advantages. It allows the use of an efficient extended Thomas algorithm, rather than a pentadiagonal solver or a sparse solver required if

more than four points are used for the approximation. There is one special case of this approximation, $\gamma = \sqrt{2}$, that is interesting in that it yields a third-order approximation, as found by Martínez-Ortiz [385]. That author also derived some conveniently compact specific approximation formulae for the exponentially expanding grid, for most cases of interest, obviating the need for a numerical computation of the approximation coefficients. The value $\sqrt{2}$ may appear a little large but if only a few points and a very small first interval are wanted, it might be useful. One would then have to find the first interval that satisfies the γ value and the desired number of points in the space region, and this can be done by simple application of (7.19). As an example, if we want $N = 14$ and $X_{lim} = 6$, this makes $H_1 = 0.0032615$.

7.2.1 Choice of Parameters

For any point sequence and a given stretching function, there are several parameters to choose. One is always N, the number of points along the profile. The others are the length of the first interval H_1 or (the same thing) the position of the first point next to the electrode X_1. This might then determine the function parameter. There are situations where setting X_1 is desirable; for example, in order to achieve a certain desired accuracy in the gradient G. If this is so, in both cases of the exponentially increasing intervals function (7.17) and (7.21), the stretching parameter then needs to be searched for by numerical means. Let the largest X-value be X_L, and the number of internal points be N. We have set the wanted X_1. For exponentially expanding intervals (7.17) we then apply (7.19) and seek a γ value that satisfies it. A simple numerical (for example, binary) search finds γ. An example of such a search is shown in the function `EE_FAC` described in Appendix C. Less conveniently, one might choose X_1 and γ, and find out what N then becomes, by a simple calculation. This is deemed less likely because one would usually want to have control over the N value.

7.3 Concluding Remarks on Unequal Spatial Intervals

The question arises of how low an N value it is possible to work with and still get good results. The simulation package DigiSim due to Rudolph and Feldberg [482] routinely uses as few as 14 and is able to achieve sufficient accuracy in the current. This depends on one's definition of "sufficient". If 0.1% accuracy is wanted, about 40 points in space might be optimal.

Clearly also, in order to choose a suitable set of parameters, one must know the requirements before the simulation. If some homogeneous rate constant changes during a series of program runs (for example one in which such a rate constant is searched for), then the grid parameters should change. This makes adaptive grids more useful. These are described below.

As for the choice between direct discretisation on an arbitrarily spaced grid or the formulae for the semi-transformed or the transformed diffusion equation, the present author now inclines towards the first of these. Formulae for the derivatives on arbitrarily spaced points are given in Chap. 3 and Appendix A, and the general subroutine `U_DERIV` is referred to in Appendix C.

7.4 Unequal Time Intervals

Just as space can be divided into unequally spaced intervals, so might time also be unevenly divided. As with spatial intervals, there is the choice between discretising on an uneven time grid or using a transformation to a new time scale. Since, except for BDF methods, one usually differentiates with respect to time using only two time points (levels), transformation does not make sense here.

Unequal time intervals are especially appropriate in the simulation of pulse experiments or, in fact, any experiments in which there are potential or current jumps away from $T = 0$. Initial steps in this direction were taken by Flanagan et al. [248], Dillard et al. [212] and Nikolić [419], who used two different time intervals: largish intervals when the current does not change much, and finer intervals (1/100 to 1/9 as large) just after a pulse. Seeber and Stefani [501] used a rather complicated scheme, in which they used expanding intervals in space and direct discretisation on that grid; and in recognition of the fact that, far away from the electrode, the larger space intervals also made larger time intervals possible there, used that as well. This seems rather awkward to keep track of. Klymenko et al. [340] combined equally divided Pearson steps with exponentially expanding time steps in a simulation of double potential step chronoamperometry.

The following is often used. We choose exponentially increasing time intervals over some period τ, which may be the total simulation period, a pulse duration or (see below) a single whole time interval to be subdivided. The period is divided into M intervals of length $\delta t_k, k = 1, \ldots, M$. Assume the recursive relation

$$\delta t_k = \gamma \, \delta t_{k-1} \tag{7.22}$$

and

$$\sum_{k=1}^{M} \delta t_k = \tau \tag{7.23}$$

(note that these equations are of the same form as (7.16) and (7.17)). Peaceman and Rachford [436] used the technique for the first time in 1955, in their classical paper describing the method of alternating directions implicit method (ADI, Chap. 12). Lavaghini et al. [355], and later Feldberg and Goldstein [236] as well as Svir and coworkers [340, 538, 540, 542], used exponentially expanding time intervals in electrochemistry.

There is no problem with varying time intervals with two-level simulation methods, but with a method like BDF, there is the problem that one needs multipoint time derivatives calculated from unequally spaced points in time. Feldberg and Goldstein [236] show how to do this and even show how to apply the Feldbergian correction of half a time interval in this case, that becomes necessary when using the simple start for BDF, described in Chap. 4 (see also the consistency proof for this procedure in Appendix B).

7.4.1 Implementation of Exponentially Increasing Time Intervals

A special case of exponentially increasing intervals, applied only to the subdivision of the first time interval, was suggested by Mocak et al. [402] and used by Britz and Østerby [148]. It is interval doubling, or the case $\gamma = 2$. In [148], the sequence for M such steps was the sequence of fractions $2^{-M+1}, 2^{-M+1}, 2^{-M+2}, 2^{-M+3}, \ldots, \frac{1}{2}$. In other words, the smallest fraction was applied twice. The general formula using (7.22) and (7.23) is implemented in a different way. This was done in a recent paper [149], using subdivision of the first time interval, in order to damp the oscillations often produced by the Crank-Nicolson method (Chap. 8). The form of the equations for the required parameters (M, γ, size of the first subinterval) is exactly like that for exponentially increasing spatial intervals, (7.16)–(7.19).

7.5 Adaptive Interval Changes

The most flexible strategy is to adapt intervals, in space or in time, according to the need at any particular time during the simulation. Ablow and Schechter refer to *campylotropic* or *curvature-seeking* coordinates [27]. It was noted above that fixed unequal spatial intervals might not be suitable if, for example, a reaction layer becomes too thin even for the first few intervals to lie within it. Worse still, the method described above, in which points are most closely spaced near the electrode, cannot accommodate sharp changes in concentration changes that occur away from the electrode, as can indeed happen. Bieniasz [97] has described a system involving a second-order homogeneous chemical reaction, in which a sharp concentration peak appears in the solution for one of the species. Only adaptive techniques can handle this situation. We distinguish between adaptation of spatial and temporal intervals. There is a vast numerical literature on this topic, and just some selected citations are given here. Bieniasz wrote a series of articles in which he introduced the idea to electrochemical simulations. In [93, 95], he described the use of a fixed number of grid points, moved about as required. He then [97] applied this to a concentration hump as mentioned above. Bieniasz later turned to a different technique [101] which starts with a coarse but evenly spaced grid, to which new points are added (and perhaps removed again later)

midway between existing points, as required. He also applied time-step adaptation [96]. Nann and Heinze [407,408] meanwhile developed a finite element method in which points (nodes) are added where needed, an idea carried forward, refined and applied to two-dimensional systems by Harriman et al. [287, 288, 289, 290, 291]. Time step adaptation, a standard in the literature on *ode*s (see such texts as [130, 284]), was first applied by Bieniasz [96] to electrochemical simulation (see below). All the references cited here include citations of the important works within the larger numerical literature.

7.5.1 Spatial Interval Adaptation

The single reference to Thompson's survey [547] must suffice to represent the numerical literature, and the references in the papers of Bieniasz [93, 95, 97, 101] provide further background.

Bieniasz began his series with an exploration of moving grids [93,95], using a fixed number of points. As a given simulation develops, the program determines whether the spacing needs to be closer or wider across the concentration profile, in a preliminary forward step, and then adjusts the point positions. This is called regridding. The criterion for moving the points is a sensitive issue, on which there is some disagreement in the literature. The essence of all schemes is to produce a so-called monitor function [122] or a function based on it, that in some way resembles the simulated variable's profile, and then to slice this into equal vertical intervals, producing new points along X which then, hopefully, place points where they are most needed. Dorfi et al. [213] suggest using a monitor function such as

$$M(i) = \sqrt{\alpha + (du/dx)^2} \tag{7.24}$$

at every point i, where u is the variable to be computed. The value of α is given as unity in older papers such as that of Blom et al. [122], but Bieniasz found [93,95] that a smaller value like 0.0005 is better in the present context. This monitor function is now integrated with respect to x to produce the monitor profile, generally given the symbol ξ:

$$\xi(x) = \frac{\int_0^x M(x)\,dx}{\int_0^{x_L} M(x)\,dx} \tag{7.25}$$

(note that it is normalised by its value at the outer limit for x, x_L, so that it rises to unity). An algorithm is then applied to it to slice it into equal vertical intervals and to find the x-positions that correspond to them. The function (7.24) is also referred to elsewhere [495, 547], to name just a few references. Blom [122], on the other hand, recommends the use of the second derivative,

$$M(i) = \sqrt{\alpha + |d^2u/dx^2|)} \tag{7.26}$$

and Bieniasz follows this suggestion [93, 95]. The reason is (private communication, Bieniasz 2001) that the first derivative is not itself of great significance, if the second derivative is small in the diffusion equation, so the second derivative indicates places in the profile where things are changing.

The procedure is then as follows. At a given time, a trial step is taken to the next time level. This produces a provisional new concentration profile. From this, the ξ-function (7.25) is generated and from it, a new set of positions for the points. Now the concentrations are interpolated at these points, between the present concentration points, and the step to the next time level repeated on the new set of points.

Let us provide an example. We take a uniform grid of just 20 points in the range $0 \leq X \leq 6$ and assume a Cottrell concentration profile at time $T = 0.5$ shown in Fig. 7.2. This is a little artificial, as one would never carry out such a drastic regridding, but it will illustrate the method better than what usually happens (small changes over a given time interval). It amounts to taking a huge step of 0.5 in T and somehow having obtained a rather accurate new concentration profile. Against the advice of Blom [122] and Bieniasz [93, 95] (see below), we compute second derivatives of the profile at all the node points, using the usual central three-point formula, except at the electrode, where an asymmetric three-point formula is used (see Sect. 3.8, page 44 and Appendix A). These are all first-order accurate if the intervals along X are not equal and we obtain the function $M(X)$ in Fig. 7.2, plotted point to point. It is integrated to $\xi(X)$ using the trapezium method. Normalisation is to its final value, at $X_L = 6$. $\xi(X)$, ranging from zero to unity, is now inverted to $X(\xi)$ and 20 X-values found for it at equal intervals in ξ of 0.05 by interpolation. Blom et al. [122] and Bieniasz [93, 95] cite a paper by de Boor [195] for this process, but it is in fact not very complicated to implement using a standard interpolation routine. Now, if one were to go on, as one normally would, a new concentration profile at the new set of positions

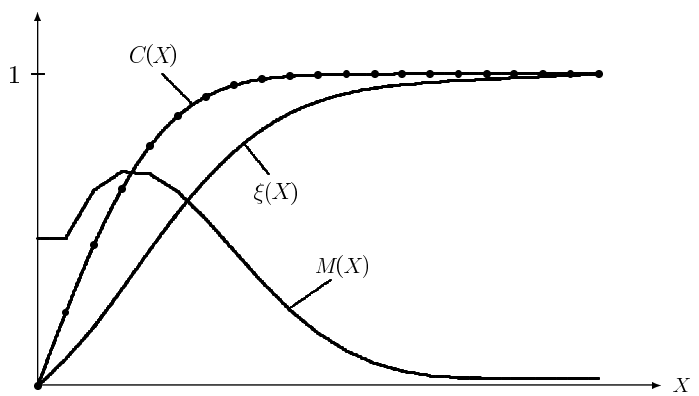


Fig. 7.2. Some profiles derived from a Cottrell profile at $T = 0.5$

along X at T is computed, again by interpolation. The shift in positions is indicated in Fig. 7.3 and it is seen that there is now a wide spacing at the far end, and a crowding of points, not near the electrode but some distance away from it, where the monitor function is maximum. That is also roughly where the greatest changes in concentration occur during the next time step; this supports the argument in favour of the second derivative in the monitor function. Note that the 20 points here are a rather small number, chosen to make the figure clearer. Bieniasz normally uses about 50, so that the possibly excessive spacing at the far end would not be so wide. Also, to some extent the wide spacing is a result of the large step in time taken in the example. A second regridding on the new grid shown in these figures does in fact lead to a smaller gap at large X and this is what one would obtain if a number of smaller steps had been taken.

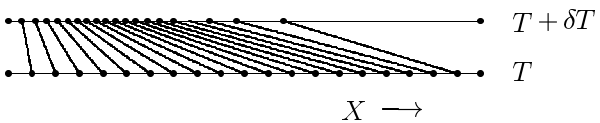


Fig. 7.3. Regridding for the Cottrell profile in Fig. 7.2

Some remarks are in order, starting with the purpose of the α term in (7.26). As mentioned, the numerical literature appears to prefer it to be close to unity. If one were to set it to zero, one would obtain an unacceptably wide spacing at parts of the profile where the second derivative is close to zero. In effect, a finite α value ensures a finite positive gradient of $\xi(X)$ at large X. If this is not done, the plateau obtained means excessively large intervals in this region upon regridding. As mentioned, in the work of Bieniasz, a value of 0.0005 was found optimal. Secondly, there is the question of how to compute the second derivatives over unevenly spaced points. Blom et al. [122], followed by Bieniasz [93], used a somewhat awkward method. The first point monitored (using their method) lies at the middle of the first interval, and the formula given by Blom et al. [122] is in fact incorrect. Presumably, the second derivative at $X = 0$ is assumed zero, which it need not be. In our example, it was computed as the one-sided three-point approximation at $X = 0$, which seems to make more sense. From there on to the far edge, Blom et al. use four-point expressions, centred on the middle of the mid-interval; however, their expression again is incorrect, rendering the use of four points useless. The object was to achieve a better approximation to the derivative, that might be second-order in the interval lengths in some sense. The expression was later corrected by Bieniasz [95]. Both teams then use the simple three-point approximations for the final calculation, presumably in order to avoid yet further interpolations. It appears that one might as well use three-point formulas in both phases, as was done in the example above. Alternatively,

one might use higher-order formulas, especially for the diffusion step, on the unequal grid, using more than three points, for example five, centered on existing points. This has not been attempted to date. Such formulae are provided in Chap. 3; a few cases are also given in Appendix A, and a general procedure for them is described in Appendix C.

Despite the fact that adaptive gridding seems to work very well (with some refinements described in [95]), being for example, until recently, the only method capable of adapting to a narrow concentration hump away from the electrode [97], Bieniasz has recently concluded [101, 102, 103, 115] that another method is better. The problems he noted are, among others, the need to set α to some value, and the problems arising from the approximation to the second derivative on an unequally spaced grid. The new method is called patch-adaptive, and works with a continually varying number of points. It is based on older work in the numerical literature (see [101] for a large number of citations). One begins with a coarse grid, and does a calculation to the next time level. This is then repeated on a grid of twice as many points, the new points placed exactly midway between the first set. This ensures a locally equal spacing and thus second-order second derivatives. The two solutions are then used to provide an error estimate. The way this is done depends on the simulation algorithm. One way might be to use extrapolation, described in Chap. 9, which can provide an error estimate. At those places along X where this error exceeds some set value in magnitude, new points are then placed midway between the existing points, and the calculation repeated. If there appear sharp gradients in the profile, more and more points will thus be inserted. All the time, however, one is working (locally) with equally spaced points and thus second-order second derivatives. The disadvantage is that one must keep track of a changing number of points along X, as points are added and perhaps removed later. This requires data structures that are not trivial to program. It seems to this author, that this renders the method less interesting to the programming electrochemist. It might be of more interest to programmers of general simulation packages.

7.5.2 Time Interval Adaptation

Just as sharp changes in the space direction point to changes in spatial intervals, so sharp changes in time demand time interval adaptation. This is in fact standard procedure in the *ode* world since the paper by Douglas [214], see such standard works as [130,284] for example. In electrochemical simulations, there have been relatively few attempts to do this. The impetus for varying time intervals comes from two problems. One problem is that of pulse techniques, especially current reversal or potential double pulses. Clearly, there are sharp changes in concentration profiles at the onset of each pulse. Crude beginnings of this [212, 248, 419], using alternately larger and smaller time intervals before and after a pulse, have been mentioned above.

Once again, an adaptive technique might be the universal answer to these problems, especially since there might be unforeseen changes at various stages during a simulation, as can happen in linear sweep voltammetry. Such a scheme has been devised by Bieniasz [96]. Upon first considering this, one might assume, say, that current changes themselves could be the factor that decides the length of the next time interval to be used. However, as with adaptive spatial intervals, this is not as good as using a kind of second derivative, for similar reasons. If there were changes in concentrations linear in time, then no matter how large these changes are, large time intervals can be used; but if the changes are themselves changing (that is, there are significant second derivatives with time), the intervals must be reduced. The picture is complicated by the fact that this will mostly be used in conjunction with adaptive spatial grids, making the second derivatives less straightforward to express. Bieniasz suggests the use of the following quantity as a kind of monitor function. Assume that a tentative step of δT has been taken on the present grid, and that a given point indexed i along X has just been moved by an amount δH; the estimate function EST is then

$$EST = \frac{\delta T^2}{2}\frac{\partial^2 C}{\partial T^2} + \delta T \delta H \frac{\partial^2 C}{\partial X \partial T} + \frac{\delta H^2}{2}\frac{\partial^2 C}{\partial X^2} \tag{7.27}$$

where the second derivatives must be discretised by some suitable expression. The present author regards this as more complicated than the average electrochemist is willing to program, and the method will not be detailed any further. It did produce impressive results [96] with square wave simulations.

There are some simpler strategies that might do, and are easier to program. If an experiment such as double pulse or square wave voltammetry is simulated, the sharp changes occur at predictable times, and simple sequences of time intervals, such as exponentially expanding intervals, can be satisfactory, repeating the sequence at the onset of each pulse.

If there are unpredictable changes, the answer might be to use a professionial package; that is, either a simulation package (see Chap. 16), or the method of lines (Chap. 9) and a professional routine for solving the resulting set of *ode*s, making use of the adaptive time intervals feature, which these routines normally offer.

8 The Commonly Used Implicit Methods

Essentially, only two implicit methods will be described here, but with extensions that make them more useful. They are derived from the implicit methods described for *ode*s in Chap. 4, BI and the trapezium method. These have different names in the *pde* context, as will be seen.

Implicit methods have the great advantage of being stable, in contrast with the explicit method. It will be seen (and analysed in detail in Chap. 14) that the Laasonen method, a kind of BI, is very stable and responds to sharp transients with smoothly declining (but relatively large) errors, whereas Crank-Nicolson, also nominally stable, responds with error oscillations of declining amplitude, but is highly accurate. The drawbacks of both methods can be overcome, as will be described below.

First, the discretisation of the second, spatial derivative of concentration will be reiterated in a general form that can then be built into the methods to follow. For the three concentrations grouped around the one at the point X_i, we can write the general linear expression,

$$\frac{\partial^2 C_i}{\partial X^2} \approx \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1} \tag{8.1}$$

in which the α coefficients are defined according to whether equal or unequal intervals are used. The three concentrations are situated at the three corresponding positions X_{i-1}, X_i and X_{i+1}. For equal intervals H in X, the coefficients are

$$\begin{aligned} \alpha_1 &= 1/H^2 \\ \alpha_2 &= -2/H^2 \\ \alpha_3 &= 1/H^2 \end{aligned} \tag{8.2}$$

as already given in Chap. 3 (3.41) and independent of the index i. If unequal intervals are used, the coefficients are

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 119–143 (2005)
www.springerlink.com

$$\alpha_1 = \frac{2}{(X_i - X_{i-1})(X_{i+1} - X_{i-1})}$$
$$\alpha_2 = -\frac{2}{(X_i - X_{i-1})(X_{i+1} - X_i)} \,. \tag{8.3}$$
$$\alpha_3 = \frac{2}{(X_{i+1} - X_i)(X_{i+1} - X_{i-1})}$$

These α's are dependent on the index i but for brevity, this will not be indicated in what follows below.

If transformation is to be used, by the function (7.3) on page 105, then the resulting right-hand-side of the diffusion (7.5) can be written as a transformation of the second spatial derivative,

$$\frac{\partial^2 C}{\partial X^2} = a^2 e^{-2Y} \left(\frac{\partial^2 C}{\partial Y^2} - \frac{\partial C}{\partial Y} \right) \tag{8.4}$$

and the (obvious) discretisation in terms of equal intervals δY then results again in a linear expression like (8.1), with the coefficients at the point Y_i given by

$$\alpha_1 = \left(1 + \frac{\delta Y}{2} \right) w_i$$
$$\alpha_2 = -2\, w_i \tag{8.5}$$
$$\alpha_3 = \left(1 - \frac{\delta Y}{2} \right) w_i$$

the common w_i being

$$w_i = \frac{a^2 \exp(-2i\delta Y)}{\delta Y^2} \,. \tag{8.6}$$

We thus have, for the diffusion equation

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} \,, \tag{8.7}$$

a suitable discretisation for all three cases with their respective definitions of the α coefficients, in the form

$$\frac{\partial C}{\partial T} = \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1} \,. \tag{8.8}$$

This will be the basis for what follows. The diffusion equation, discretised on the right-hand side as in (8.8), is now a system of *ode*s in the concentration vector $\boldsymbol{C}$, of the form

$$\frac{\partial \boldsymbol{C}}{\partial T} = f(\boldsymbol{C}) \tag{8.9}$$

and the two main implicit methods will be seen to be analogous to those used for *ode*s.

8.1 The Laasonen Method or BI

When applied to the solution of *ode*s, the BI method (Chap. 4) uses a backward difference for the derivative on the left-hand side of (8.9) and the argument of the function on the right-hand side is the future, unknown, concentration vector. In our notation, at the point i along the row of concentrations, this is

$$\frac{C'_i - C_i}{\delta T} = \alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} \; . \tag{8.10}$$

This was formulated by Laasonen [343] in 1949. Rewriting this, the equation set becomes

$$\begin{aligned}
C'_0 \quad &+ a_{1,1} C'_1 \quad + a_2 C'_2 \quad &= b_1 \\
C'_1 \quad &+ a_{1,2} C'_2 \quad + a_2 C'_3 \quad &= b_2 \\
&\cdots \\
C'_{i-1} \quad &+ a_{1,i} C'_i \quad + a_2 C'_{i+1} \quad &= b_i \\
&\cdots \\
C'_{N-1} \; &+ a_{1,N} C'_N \; + a_2 C'_{N+1} \; &= b_N
\end{aligned} \tag{8.11}$$

with the coefficients given by

$$\begin{aligned}
a_{1,i} &= \frac{\alpha_2 - 1/\delta T}{\alpha_1} \\
a_2 &= \frac{\alpha_3}{\alpha_1} \\
b_i &= \frac{-1}{\delta T \alpha_1} C_i
\end{aligned} \tag{8.12}$$

in which the coefficient a_2 has been written as independent of i. In most cases, it will in fact be constant, and equal to unity for equal intervals in X. For exponentially expanding intervals as in (7.19), α_3/α_2 is equal to $1/\gamma$, that is, the inverse of the interval expansion factor. Also, the α coefficients, as given above for the various cases of equal, unequal or transformed intervals, are all i-dependent except in the case of equal intervals, which is the reason that a_1 and the factor in b_i are so, as well.

The solution of the above system of (8.11) will be described below, together with that for the CN method.

8.2 The Crank-Nicolson Method, CN

This method derives from the trapezium method in the *ode* field in which the time derivative in (8.9), expressed exactly as in (8.10), becomes a second-order central difference by virtue of the fact that the right-hand side now

refers to a point in time midway in the time interval. This is achieved by taking the average of the second spatial derivative at the present time T and that at $T + \delta T$:

$$\frac{C'_i - C_i}{\delta T} = \frac{1}{2}\left(\alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} + \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1}\right) . \tag{8.13}$$

The result is a system exactly as (8.11) but with different definitions of the coefficients:

$$\begin{aligned} a_{1,i} &= \frac{\alpha_2 - 2/\delta T}{\alpha_1} \\ a_2 &= \frac{\alpha_3}{\alpha_1} \\ b_i &= -C_{i-1} - a_{3,i} C_i - a_2 C_{i+1} \end{aligned} \tag{8.14}$$

and the new coefficient

$$a_{3,i} = \frac{\alpha_2 + 2/\delta T}{\alpha_1} . \tag{8.15}$$

Again, a_1 and a_3 are dependent on the index i, by virtue of the fact that the α's are i-dependent.

Crank-Nicolson bears the name of its inventors [185]. It is interesting to note that in their paper, they cite Hartree and Womersley [296], who describe what amounts to its precursor.

8.3 Solving the Implicit System

The system (8.11) shown above, that is the result of discretising either according to the Laasonen or CN method, can be solved efficiently by the Thomas algorithm [546]. This recognises that the system is tridiagonal. It can be reduced to a didiagonal system by working from either end, that is, from C'_0 or C'_{N+1}. The latter approach is better here. The last equation in (8.11) has a term in C'_{N+1}, which is the bulk value, not subject to diffusional changes, being a boundary value. Normally, it is constant, equal to the initial bulk value. In some cases, it can change with time, for example in the Reinert-Berg [464] or the Birk and Perone [121] systems, in which the reacting substance itself undergoes a homogeneous decay reaction. In such cases, the value of C'_{N+1}, while not constant, is still accurately predictable and thus known at any time. Thus, this known term can be moved to the right-hand side of the last equation of the system, thus giving the new last equation

$$C'_{N-1} + a_{1,N} C'_N = b_N - a_2 C'_{N+1} \tag{8.16}$$

with only two unknowns. This is rewritten in the form

$$C'_{N-1} + a'_N C'_N = b'_N \tag{8.17}$$

with, clearly,

$$a'_N = a_{1,N} \tag{8.18}$$

and

$$b'_N = b_N - a_2 C'_{N+1} \, . \tag{8.19}$$

Equation (8.17) is now used to express C'_N in terms of C'_{N-1}:

$$C'_N = \frac{b'_N - C'_{N-1}}{a'_N} \tag{8.20}$$

and this is substituted into the second-last equation of the system (8.11),

$$C'_{N-2} + a_{1,N-1} C'_{N-1} + a_2 C'_N = b_{N-1} \tag{8.21}$$

giving, after some tidying up, the next new equation

$$C'_{N-2} + a'_{N-1} C'_{N-1} = b'_{N-1} \tag{8.22}$$

with

$$a'_{N-1} = a_{1,N-1} - \frac{a_2}{a'_N} \tag{8.23}$$

and

$$b'_{N-1} = b_{N-1} - a_2 \frac{b'_N}{a'_N} \, . \tag{8.24}$$

This process continues in the backward direction and the recursive expressions for the coefficients in the i^{th} equation generated,

$$C'_{i-1} + a'_i C'_i = b'_i \tag{8.25}$$

are

$$a'_i = a_{1,i} - \frac{a_2}{a'_{i+1}} \tag{8.26}$$

and

$$b'_i = b_i - a_2 \frac{b'_{i+1}}{a'_{i+1}} \tag{8.27}$$

(starting with (8.18) and (8.19)) until the first equation is reached,

$$C'_0 + a'_1 C'_1 = b'_1 \, . \tag{8.28}$$

At this point, we have a new system of equations, each with two unknowns. The point of attack now is C'_0, the boundary value. How this is calculated, has been described in Chap. 6. When this is done, the process goes forward again, solving explicitly for all unknowns, starting with

$$C'_1 = \frac{b'_1 - C'_0}{a'_1} \tag{8.29}$$

or, for a general C'_i,

$$C'_i = \frac{b'_i - C'_{i-1}}{a'_i} \ . \tag{8.30}$$

In Appendix C, a few examples of the use of CN are described: for a Cottrell simulation (`COTT_CN`), chronopotentiometry (`CHRONO_CN`) and LSV (`LSV_CN`).

8.4 Using Four-Point Spatial Second Derivatives

It was shown in Chap. 7 that the three-point second spatial derivative on an unequally spaced grid, leading to (8.1) with the coefficients defined in (8.3), can be improved with relatively small effort to an asymmetric four-point formula, spanning the indices $i-1, i, i+1, i+2$, with the second derivative referred to the point at index i. The diffusion equation is then semi-discretised to

$$\frac{dC}{dT} = \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1} + \alpha_4 C_{i+2} \tag{8.31}$$

analogous to the three-point form (8.8). The derivation of the coefficients are described in Chap. 3, and some formulae given in Appendix A, and a procedure (`U_DERIV`) described in Appendix C. The above α values are of course i-dependent. Here, we describe only the implementation of the scheme to the Laasonen method, leaving out CN. The reason is that the Laasonen method best enables the use of extrapolation, of which the simple second-order variant nicely couples with the second-order four-point approximation. Thus, using Laasonen, (8.31) becomes

$$\frac{C'_i - C_i}{\delta T} = \alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} + \alpha_4 C'_{i+2} \tag{8.32}$$

which leads to the new system of equations

$$\begin{array}{lllll}
C'_0 & + a_{1,1}C'_1 & + a_{2,1}C'_2 & + a_{3,1}C'_3 & = b_1 \\
C'_1 & + a_{1,2}C'_2 & + a_{2,2}C'_3 & + a_{3,2}C'_4 & = b_2 \\
\ldots & & & & \\
C'_{i-1} & + a_{1,i}C'_i & + a_{2,i}C'_{i+1} & + a_{3,i}C_{i+2} & = b_i \\
\ldots & & & & \\
C'_{N-2} & + a_{1,N-1}C'_{N-1} & + a_{2,N-1}C'_N & + a_{3,N-1}C'_{N+1} & = b_{N-1} \\
C'_{N-1} & + a_{1,N}C'_N & + a_{2,N}C'_{N+1} & + a_{3,N}C'_{N+2} & = b_N
\end{array} \tag{8.33}$$

with the coefficients given by

$$\begin{aligned} a_{1,i} &= \frac{\alpha_2 - 1/\delta T}{\alpha_1} \\ a_{2,i} &= \frac{\alpha_3}{\alpha_1} \\ a_{3,i} &= \frac{\alpha_4}{\alpha_1} \\ b_i &= \frac{-1}{\delta T \alpha_1} C_i \ . \end{aligned} \tag{8.34}$$

Note that this differs from the three-point system (8.11) in that every line now has four coefficients, and their definitions have been written as i-dependent, which is the case for exponentially expanding X positions. Note also that an extra point, index $N+2$, has been added to the row of X values. Point N is still the last one to undergo diffusional changes, so the fact that the point $N+2$ lies rather far past X_{lim} (which is at index N), does not matter, both these extra points being either constant or set values, not subject to diffusional changes.

The above system, although leading to a quadradiagonal system of equations, can still be solved by a smallish extension of the Thomas algorithm [153]. Consider the last two equations of (8.33) and rewrite them, putting the bulk concentration terms on the right-hand side:

$$\begin{aligned} C'_{N-2} + a_{1,N-1} C'_{N-1} + a_{2,N-1} C'_N &= b_{N-1} - a_{3,N-1} C'_{N+1} \\ C'_{N-1} + a_{1,N} C'_N &= b_N - a_{2,N} C'_{N+1} - a_{3,N} C'_{N+2} \ . \end{aligned} \tag{8.35}$$

We rewrite the last equation, which is already down to two unknowns, in the form

$$C'_{N-1} + a'_N C'_N = b'_N \tag{8.36}$$

(a'_N and b'_N being obvious, and defined below in (8.38)) which, as before, allows the substitution for C'_N in the next-last equation, which then also reduces to two unknowns,

$$C'_{N-2} + a'_{N-1} C'_{N-1} = b'_{N-1} \ . \tag{8.37}$$

Thus far, this looks just like the Thomas algorithm for the tridiagonal system, as described above in Sect. 8.3. From here on, however, the processes diverge. We need to keep both substitutions for C'_N and C'_{N-1} and use them in the third-last equation, which contains both. This process is continued backwards, reducing all equations with four unknowns to new ones with just two unknowns. The expressions resulting from this are the following:

$$a'_N = a_{1,N}; \qquad b'_N = b_N - (a_{2,N} + a_{3,N}) C_b \ , \tag{8.38}$$

C_b being the bulk concentration, equal to C'_{N+1} and C'_{N+2} in system (8.33). Then,

$$a'_{N-1} = a_{1,N-1} - \frac{a_{2,N-1}}{a'_N} \; ; \qquad b'_{N-1} = b_{N-1} - a_{2,N}\frac{b'_N}{a'_N} - a_{3,N-1}C_b \; . \quad (8.39)$$

These now serve as starting values for the recursive process; the ith equation of system (8.33) becomes

$$C'_{i-1} + a'_i C'_i = b'_i \quad (8.40)$$

with the two new coefficients recursively given by

$$a'_i = a_{1,i} - \frac{1}{a'_{i+1}}\left(a_{2,i} - \frac{a_{3,i}}{a'_{i+2}}\right) \quad (8.41)$$

$$b'_i = b_i - a_{2,i}\frac{b'_{i+1}}{a'_{i+1}} - a_{3,i}\left(\frac{b'_{i+2}}{a'_{i+2}} - \frac{b'_{i+1}}{a'_{i+1}a'_{i+2}}\right) \; . \quad (8.42)$$

This is not so difficult to program and leads to a new system just like that in Sect. 8.3, right down to (8.28). Boundary condition handling is the same, as is the forward scan that yields all the new unknowns, (8.30).

This has been programmed into example program `COTT_EXTRAP4` described in Appendix C. Compared with the three-point program, `COTT_EXTRAP`, it yields results, using the same parameters, with an accuracy about an order of magnitude better. Once programmed and debugged, the code can be easily transplanted into other programs and seems worthwhile implementing.

Finally, as mentioned earlier (Chap. 7, Sect. 7.2), Martínez-Ortiz [385] developed some rather simple formulae for derivative approximations for the special case of exponentially expanding grid spacings, and in the course of this work discovered that the four-point second-order derivative approximation $u''_2(4)$, for the expansion factor $\gamma = \sqrt{2}$ is third-order in accuracy, rather than second, as it is for other γ values. This could be an easy and useful way to increase the accuracy, using the four-point formula.

8.5 Improvements on CN and Laasonen

The Laasonen method, because of the forward difference in T, has errors of $O(\delta T, H^2)$, and the first-order behaviour with respect to δT limits its accuracy to about the same as the explicit method described in Chap. 5. However, it has a smooth error response to disturbances such as an initial transient (Cottrell), and is stable for any value of $\delta T/H^2$, where H is either the same as all intervals if equal intervals are used in X, or is the smallest (usually the first) interval if unequal intervals are used. This makes the method interesting, and it will be seen below that it can be improved. For simplicity, the symbol λ will be used below, and denotes the largest value of that parameter, that is, the value from the smallest interval in space in a given system.

CN is formally as stable as Laasonen, and more accurate, with errors of $O(\delta T^2, H^2)$. However, it has one serious drawback. If the initial conditions are a sharp change in concentration (as in potential jump experiments), CN responds with errors oscillating about zero and for large λ values these oscillations can persist over much of the simulation period. This has meant that simulators have tended to use other methods instead. The stability, and the reason for the oscillatory response, of CN are explained in Chap. 14, but here, a method of damping the oscillations will be described.

To appreciate the problems with both CN and Laasonen, consider Fig. 8.1. This shows three curves, and we ignore the stippled one for the moment. The plot shows simulations of the Cottrell system, using only 20 steps in the range $0 < T \leq 1$ and a λ value of 3 (equal spatial intervals). The vertical axis is the relative error in the computed current. The smoothly falling solid line is that for the Laasonen method, while the oscillating solid line is from CN. The plot does not make clear that the error at $T = 1$ is in fact greater for Laasonen than for CN, but it does illustrate the problem of oscillation with CN. To show the difference in final errors, Fig. 8.2 shows the same results but with the vertical scale now narrowed down. At smaller times, the plot lies outside the range, but now it is clearly seen that although the CN curve is still oscillating, the Laasonen response has the greater error at $T = 1$.

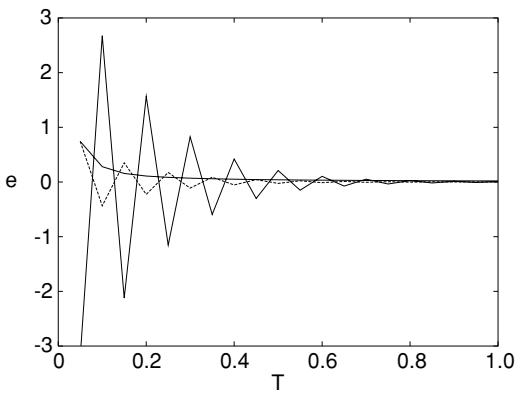


Fig. 8.1. Relative error in computed current vs time

8.5.1 Damping the CN Oscillations

Crank and Nicolson, in their original paper [185], recognised the oscillation problem with their method, writing "If γ [which is their $\delta t/\delta x^2$] is very large an oscillatory error which only disappears very slowly may arise". The problem is referred to in most texts describing the method. More detail is given in Chap. 14, but the essence of the problem is that CN will oscillate if $\lambda > 0.5$;

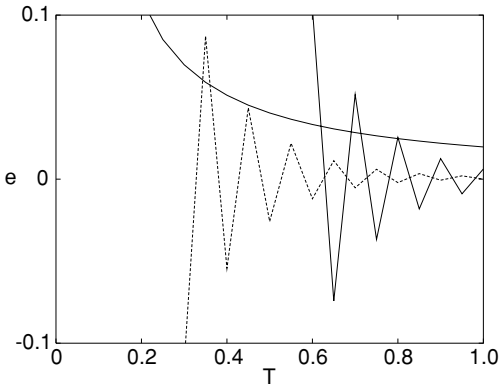


Fig. 8.2. Relative error in computed current vs time, narrow scale

in practice, a value of unity or even 2 will not cause serious oscillations. So one way to reduce oscillations is to lower λ, usually by reducing δT. This might cause unduly long execution times, but fortunately, once oscillations have been damped, they normally do not reappear, so if they can be damped within the first interval, the problem is solved. This leads to the most effective method, subdivision of the first time interval.

First-Interval Subdivision

There are a few ways of subdividing the first time step: using a number M of equal intervals, or a number of intervals expanding with time, usually exponentially. The two methods can be formally combined into one description. Let the full interval to be subdivided be of length δT, and let it be subdivided into M smaller intervals $\tau_i, i = 1 \ldots M$, such that

$$\sum_{i=1}^{M} \tau_i = \delta T \tag{8.43}$$

and

$$\tau_i = \tau_{i-1}\gamma = \tau_1 \gamma^{i-1} \ . \tag{8.44}$$

Exponentially expanding subintervals are obtained if $\gamma > 1$ and equal intervals for $\gamma = 1$. The latter method is (here) called **Pearson**, after the author who first suggested it [437], in 1965. It has been studied more recently [149,228,431]. Exponentially expanding subintervals will be called **ees** here. They were suggested [148] and later used [138, 265, 340], and studied in some detail recently [149, 431].

Whether Pearson (in the one form or the other) or ees is to be used, is a matter of taste. Pearson is the simpler method, and it is simpler to determine the only parameter involved, M. Numerical experiments [149] show that a

(sub)λ value of about unity is sufficient to damp oscillations during the first M substeps, so this sets M simply to

$$M = \lambda^{-1} \tag{8.45}$$

rounded to the nearest integer (and the sub-λ then recalculated - it might not be exactly unity!). Besides simplicity, this has the additional advantage of equal time intervals: in many simulations, the coefficients as in (8.14) depend on the time interval, and must be recalculated if that interval varies. They must thus be calculated once, prior to the M sub-steps, and again once upon resumption of the full δT. With unequal intervals, however, they must be recalculated before each substep. In some cases, this can have serious effects on computing time, even though in principle, fewer substeps can be used if they are expanding. The simple Pearson start is automatically used in the example program `COTT_CN` (Appendix C), and also, although as it turns out, not really necessary, in `CHRONO_CN` (chronopotentiometry does not cause oscillation problems with CN). If a large λ value is used, Pearson can result in an excessive number of substeps, and ees will then be better.

For ees, there is no simple recipe for the choice of M and γ. The reader is referred to the study [149], where several contour plots are provided that can help. A rough guide is that $\gamma = 1.5$ is a fairly universally useful value. It is the opinion of the present author that Pearson is the best choice here.

If ees is considered desirable, there is the small matter of the determination of the parameters. Normally, one would choose M first, and then either the size of the first interval τ_1 (which sets the expansion parameter γ), or γ, which sets the first interval. In the former case, having chosen M and τ_1, the function `EE_FAC` (see Appendix C) can then be used to find the appropriate γ. In the latter case, (7.19) on page 108 can be inverted to give explicitly

$$\tau_1 = \delta T \frac{\gamma - 1}{\gamma^M - 1} \; . \tag{8.46}$$

Finally, there is another mode of operation for ees. If one considers the total simulation time as a single step, this can be subdivided into a number of exponentially expanding "subintervals" in the same manner as the above description of subdivision of the first interval. This was first suggested by Peaceman and Rachford in 1955 [436], in their famous paper describing the ADI method (see Chap. 12), and was used later [236, 355]. It is routinely used by Svir and coworkers [538, 540]. These workers tend to use strong expansion with $\gamma = 2$, which has been found not to be optimal [149]. The method requires a large number of recalculations of the coefficients and thus uses more computer time than equal intervals with a damping device applied to the first interval.

Initial BI Step(s)

Rannacher and coworker [374, 461, 462] have experimented with the idea of starting a CN simulation with one or more BI steps. The rationale is that BI damps errors in a non-oscillatory manner, unlike CN, and in fact the larger λ is, the more strongly an error is damped. This also applies to an initial transient or singularity, as encountered in a potential jump, for example. The disadvantage of BI, used for a whole simulation, is that it has a global first-order error with respect to the time interval. However, the local error for each individual step is second-order and it turns out that if one uses a fixed number of BI steps to start with and then continues with CN, the global error is still second-order. Rannacher and his coworker seem to prefer taking two or even four initial BI steps. In their second 1982 paper [374], they hint at a problem with the larger number of steps and that two might after all be preferable. However, by 1984, Rannacher again preferred four. The problem is that although the global error order does indeed remain $O(\delta t^2)$ for any fixed number M of BI steps, the error itself increases with M. If the method is used to damp oscillations with CN, then there must be a compromise between the degree of damping and the acceptable error. The method has been studied again recently by Khaliq and Wade [337], who also advocate taking four BI steps, and more recently by others [149,431,432], in which a single BI step was investigated, among other methods. Some tests convince the present author, that a single step is probably the best or at most two, and if that does not remove the oscillations sufficiently, another method should be used, such as Pearson.

The method works well if $\lambda \gg 1$ or, in the case of unequal intervals or two-dimensional geometries, where there is some critical, largest effective λ greatly exceeding unity. It was found [149] that the method works very well with a single BI step in the case of (2-D) microdisk simulations, where indeed large effective λ values result at the disk edge and it is these that are responsible for the oscillations if CN is used.

The dotted line in Figs. 8.1 and 8.2 is the response for a single Laasonen step followed by CN from then on. For this simulation, λ was set at 3, not a very large value. Nevertheless, the single Laasonen step has clearly reduced the oscillation amplitudes.

As a final note on this method, it is worth noting that Wood and Lewis [575] essentially used this method, possibly not realising it. In an investigation of how to damp CN's oscillations, one of the methods they tried was to average the initial values of the simulated quantities with the result of the first CN step. It can readily be shown that this is equivalent to taking a single BI step of half a time interval. They found some damping, but they must also have introduced an error in the time by half an interval, which would persist thereafter and degrade the accuracy, probably to first-order.

Averaging and Extrapolation

Lindberg [364] investigated smoothing of the trapezoidal response in the solution of systems of *ode*s, and this is of course a related problem. The idea of averaging is that if a sequence of errors show alternate signs (which is what we mean by oscillation), then combining several in a sequence might eliminate or reduce the error. Lindberg used a three-point averaging formula, and combined it with extrapolation (see below, and Sect. 4.9). Extrapolation alone, used with CN, does not damp oscillations but does help, marginally, in conjunction with averaging. It seems, however, that Lindberg's results do not justify these techniques.

Recommendations

A choice needs to be made. The reader may or may not want to experiment with the various possibilities. It is possible to provide some guidance here. Clearly, if λ is very large, then M can become ridiculously large if program `COTT_CN` is used – M will be equal to λ. It is in such cases, however, that the BI method works best. So a rough guide might be the following. For $3 \leq \lambda \leq 100$, use Pearson; for larger λ, the BI method might be favourable, perhaps taking 2–4 initial BI steps despite the slight loss in accuracy.

8.5.2 Making Laasonen More Accurate

In contrast to CN, Laasonen has a very acceptable error response, damping the error (and initial concentration transients) smoothly, especially at high λ; but it has the disadvantage of poor accuracy, being globally first-order with respect to δT. There are two popular ways of increasing the accuracy (raising the order) of the method, while preserving the smooth error response.

The two methods are BDF and extrapolation. Both methods are used for the numerical solution of *ode*s and are described in Chap. 4. The extension to the solution of *pde*s is most easily understood if the *pde* is semidiscretised; that is, if we only discretise the right-hand side of the diffusion equation, thus producing a set of *ode*s. This is the Method of Lines or **MOL**. Once we have such a set, as seen in (8.9), the methods for systems of *ode*s can be applied, after adding boundary conditions.

BDF

The BDF method has been described. One starts with the system such as (8.9), and goes on from there as described. This was first suggested by Richtmyer in 1957 [470], who suggested the three-point variant, and was first used in electrochemistry by Mocak and Feldberg [402] and later refined to variable time intervals by Feldberg and Goldstein [236]. These workers call it FIRM,

an abbreviation of "**F**inite **I**mplicit **R**ichtmyer **M**odification". The modification referred to is that of the BI (or Laasonen) method to a multi-level backward differentiation method. These authors also use a very simple start-up strategy, described as the **simple start with correction** in Sect. 4.8.1 on page 58. As is shown in Appendix B, this method, by good luck, provides second-order accuracy at the corrected times. The second-order nature of it also implies, however, that there is little sense in going to more than 3-point BDF. With 3 points in time, the global error is of $O(\delta T^2)$, and although an increase in the number of points included in the BDF algorithm raises that order (up to 7-point can be used, and Feldberg and coworkers now routinely use 5-point or fourth-order), this is held down to second-order by the start-up method. However, a second-order method is very useful, being of the same order as CN, and 3-point BDF has the smooth error response of Laasonen. One small drawback of the method is that additional concentration vectors must be kept in memory; in the case of three-point, one extra array is needed. This is not so bad, and the results might be considered worthwhile.

There have been attempts to improve the performance of BDF, which is normally limited by the second-order (in the spatial interval H) discretisation of the spatial derivative. Higher-order spatial second derivatives have been tried out in connection with BDF [152,154]. They can only work as intended if a high-order start is used, such as the **KW** start as described in Sect. 4.8.1. This start was not found to be efficient in [154], but it may be that a technique other than the one used there, such as Numerov (see Chap. 9), which does not produce banded matrices, will make the use of KW efficient and thus interesting. For this reason, the KW start is described below.

First, it is worthwhile detailing the implementation of BDF itself, ignoring startup for the moment. We choose 3-point BDF. Based on (4.28) given on page 57 for *ode*s, the diffusion (8.8) is discretised at index i as

$$\frac{\frac{1}{2}{}'C_i - 2C_i + \frac{3}{2}C'_i}{\delta T} = \alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} \tag{8.47}$$

where now ${}'C_i$ indicates the concentration at point i and $T - \delta T$, that is, the past concentration point. When this system of discrete equations is rearranged, it is of the same form as (8.11), with the new coefficients

$$\begin{aligned} a_{1,i} &= \frac{\alpha_2 - \frac{3}{2}/\delta T}{\alpha_1} \\ a_2 &= \frac{\alpha_3}{\alpha_1} \\ b_i &= \frac{{}'C_i}{2\delta T \alpha_1} - \frac{2C_i}{\delta T \alpha_1} \end{aligned} \tag{8.48}$$

which is seen to be hardly more complicated than those for Laasonen. There is the added step at every iteration of moving the concentration rows down

one level, that is, the rows for the present level T and $T+\delta T$ (now computed) down to, respectively, $T-\delta T$ and T.

Now for the KW start for BDF. The description in [154] will be followed here. First of all, (8.47) is rewritten in *ode* form for the whole system, replacing the left-hand side by the time derivative and the right-hand side by the general matrix form

$$\frac{d\boldsymbol{c}_j}{dt} = \frac{\lambda}{\delta t}\left(\boldsymbol{A}\boldsymbol{c}_j + \boldsymbol{s}\right) \tag{8.49}$$

where $\boldsymbol{c}_j$ refers to the whole $\boldsymbol{c}$ vector across the spatial dimension at time index j, matrix $\boldsymbol{A}$ arises from the coefficients such as those on the right-hand side of (8.47) (but may be those from any other discretisation, including multi-point or Numerov, see the next chapter), and $\boldsymbol{s}$ is the vector arising from boundary conditions. The left-hand side of this equation now leaves us free to choose the particular BDF form. As mentioned, the problem to solve here is providing the first few concentration rows. It might be thought that for k-point BDF, $k-2$ new $\boldsymbol{c}_j$ are needed, those for $j = 1 \ldots k-2$ (we already have the initial row for $j = 0$). Indeed this is true, but it turns out that the row $k-1$ is best included in the calculation, as this produces values of the same accuracy order with respect to t as the subsequent BDF steps. This was mentioned in the chapter on *ode*s, Sect. 4.10. There is then always one more equation to choose from than needed, and a choice has to be made of which ones to use. This is more or less arbitrary. For example, for 3-point BDF, we calculate rows for $j = 1$ and 2. For these, we can refer the time derivative to two out of levels 0, 1 and 2. If we choose levels 1 and 2, we have the two matrix equations

$$\frac{\boldsymbol{c}_2 - \boldsymbol{c}_0}{2\delta t} = \frac{\lambda}{\delta t}\left(\boldsymbol{A}\boldsymbol{c}_1 + \boldsymbol{s}\right) \tag{8.50}$$

referring to $j = 1$ (thus a central difference), and

$$\frac{\boldsymbol{c}_0 - 4\boldsymbol{c}_1 + 3\boldsymbol{c}_2}{2\delta t} = \frac{\lambda}{\delta t}\left(\boldsymbol{A}\boldsymbol{c}_2 + \boldsymbol{s}\right) \tag{8.51}$$

employing a BDF form referring to $j = 2$. These two equations combine into the single matrix equation

$$\begin{bmatrix} 2\lambda\boldsymbol{A} & -\boldsymbol{I} \\ 4\boldsymbol{I} & (2\lambda\boldsymbol{A} - 3\boldsymbol{I}) \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c}_0 & -2\lambda\boldsymbol{s} \\ \boldsymbol{c}_0 & -2\lambda\boldsymbol{s} \end{bmatrix} . \tag{8.52}$$

Corresponding higher-order forms can now be constructed using the examples in Chap. 4, Sect. 4.10. In the present context, the matrix equations get rather large for larger k. If there are N unknowns across the spatial dimension, then the matrix equation will be $(k-1)N \times (k-1)N$. So the method might be suitable only for smallish N.

Extrapolation

Extrapolation is described in Chap. 4, Sect. 4.9. It can easily be adapted to *pde*s, as suggested by Lawson and Morris in 1978 [356], followed by Gourlay

and Morris [278]. Strutwolf et al. [534, 535] first described its use in electrochemistry, and a higher-order variant was described later [531], in an attempt to take advantage of the higher-order extrapolation schemes. Bieniasz [100] now gives extrapolation based on BI the name **LMGE-x** (with x the order, for example LMGE-2), meaning Lawson-Morris-Gourlay extrapolation. Here it will be simply called extrapolation. The "BI" might be redundant, as it is always BI that forms the basis for extrapolation, although other methods can in principle be enhanced by extrapolation. Thus, Hartree and Womersley [296] used it in connection of their method, which has the essential elements of CN.

As with BDF, the simpler second-order scheme appears about optimal. This method also shows the same smooth and damped error response of Laasonen, with the accuracy of CN. The drawback is that for every step, several calculations must be performed – in the case of second-order extrapolation, three in all (see Sect. 4.9). This also implies an extra concentration array, for the final application of the formula, for example the vector equivalent of (4.31), requiring the result of the first, whole step, and then the result of the two half-steps. Discretisation for extrapolation is the same as for Laasonen (coefficients as in (8.12)), but using two different values of δT. There are example programs using extrapolation (`COTT_EXTRAP` and `COTT_EXTRAP4`) referred to in Appendix C.

8.6 Homogeneous Chemical Reactions

Homogeneous chemical reactions (*hcr*s) have already been mentioned in Chap. 5, where a simple explicit treatment is given. Some of the problems are also mentioned there. For the explicit method, the main one is that if a term like $K\delta T$ in the discrete equation exceeds a few percent, the simulation is inaccurate [418]. For large rate constants, this means unacceptably small δT values leading to very long computation times. Improvements were sought at the time, such as the use of Runge-Kutta integration either for just the hcr part [418] or for the whole simulation [135], and tricks such as the "sequential method" (described in Chap. 5). These did not work very well. In a previous work [134], the present author classified *hcr*s into the three categories slow, medium and fast, and for each of them a different method was suggested. Slow *hcr*s could be handled by the explicit method, medium-rate ones by implicit methods or Runge-Kutta, and fast reactions could only be handled by mathematical tricks such as that of Ruzić [483, 486], the socalled heterogeneous equivalent, in which the *hcr*s was combined with the heterogeneous electron transfer reaction into a single new one with a different (equivalent) heterogeneous rate constant.

Such tricks are no longer needed. Since the early 1990's, several advances were made in simulation that have solved all the problems, and we can now

handle all *hcr*s with efficient implicit methods in a straightforward manner. These solutions are described in what follows.

The main problems that have been solved are these:

- thin reaction layers
- nonlinear equations (and negative concentrations)
- coupled systems.

The problem of thin reaction layers are described sufficiently in Chap. 5. The solution is to use unequal intervals, that is, a few very small intervals near the electrode, so that there are sample points within the thin profile. This can be done up to a point by a fixed unequal grid such as the exponentially expanding grid described in Chap. 7. A more flexible approach is the moving adaptive grid also described in that chapter. This problem is thus solved and needs no further attention here.

In Appendix C, the program `CV_EC` is described, which simulates a CV for a simple EC reaction.

8.6.1 Nonlinear Equations

If a given *hcr* is of higher than first order, nonlinear terms arise in the dynamic equation(s). With terms, for example, in squared concentrations (see below), there is the danger, due to computational errors, that a concentration becomes negative, after which it can never be corrected. The technique CN is especially prone to this, because of the oscillations it engenders as a response to sharp transients such as a potential jump. This is one reason some workers prefer the Laasonen method or its improved offshoots, which have a smooth error response without any oscillations. With a Pearson start, however, CN can be used safely, without the appearance of negative concentrations.

Until fairly recently, the problem was regarded as too hard. For example, Fisher and Compton [245], in a study involving coupled equations with second-order terms, used explicit discretisation for the second-order terms. This degrades the accuracy of the simulation and forces very small time intervals.

With the usual nonlinear terms, which are either of the form of a squared term or the product of two species' concentrations, there are two approaches. One of them is to approximate the nonlinear terms by linearised terms. The approximations are different for CN and Laasonen.

Linearising Squared Concentration Terms

This has been mentioned in several papers, but the first to describe such approximations were Mastragostino et al. in 1968 [386]. Let the squared term be C^2; for example, a term in $-KC^2$ in the dynamic equation. The change δC is equal to $C' - C$.

For the Laasonen method, C^2 is expressed as the square of the next, unknown concentration, C'^2. We have

$$\begin{aligned} C'^2 &= (C + \delta C)^2 \\ &= C^2 + 2C\delta C + \delta C^2 \\ &\approx C^2 + 2C(C' - C) \\ &= 2CC' - C^2 \ . \end{aligned} \tag{8.53}$$

This is now a linear expression in the unknown, C', since C is known. The approximation is $O(\delta C^2)$, since a term of that order was dropped.

For CN, discretisation makes the squared term the mean of the old and new terms, so

$$\begin{aligned} \tfrac{1}{2}(C^2 + C'^2) &= \tfrac{1}{2}(C^2 + (C + \delta C)^2) \\ &\approx \tfrac{1}{2}(2C^2 + 2C\delta C) \\ &= C^2 + C(C' - C) \\ &= CC' \ . \end{aligned} \tag{8.54}$$

Again, this is $O(\delta C^2)$.

Linearising the Product of Two Species' Concentrations

For convenience, the two species' concentrations are given the symbols A and B here, with A' and B' the unknowns. Lerke et al. mention this briefly [361], for the DuFort-Frankel method [216], for some time a method suggested by Feldberg [233].

For Laasonen, we then have

$$\begin{aligned} A'B' &= (A + \delta A)(B + \delta B) \\ &= AB + B\delta A + A\delta B + \delta A\, \delta B \\ &\approx AB + B(A' - A) + A(B' - B) \\ &= A'B + AB' - AB \ . \end{aligned} \tag{8.55}$$

For CN,

$$\begin{aligned} \tfrac{1}{2}(AB + A'B') &= \tfrac{1}{2}(AB + (A + \delta A)(B + \delta B)) \\ &\approx \tfrac{1}{2}(2AB + B\delta A + A\delta B) \\ &= \tfrac{1}{2}(2AB + B(A' - A) + A(B' - B)) \\ &= \tfrac{1}{2}(A'B + AB') \ . \end{aligned} \tag{8.56}$$

This covers all the cases now.

An Example Case; Linearising

To show how this is done both in the linearised and the nonlinear form, a simple example is chosen, having the advantage of being a single-species mechanism. It is that described by Birk and Perone [121]. The electroactive substance A is formed at a uniform (bulk) concentration in a cell by a flash of light. It begins immediately to decay in a second-order *hcr*, while being electrolysed in a Cottrell-like experiment. This system will be called **BP** here. The equations are

$$\begin{aligned} A + e^- &\rightarrow B \\ 2A &\rightarrow prod \end{aligned} \tag{8.57}$$

with the chemical step irreversible and with (dimensionless) rate constant K. The normalised dynamic equation is then

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} - 2KC^2 \; . \tag{8.58}$$

The number 2 seems to be controversial but seems logical because every time two molecules of A react, both are removed from solution. Birk and Perone presented a solution for the current, but this was incorrect and was later corrected and augmented by solutions for various electrode geometries [146]. So a solution exists that can be used to test a simulation.

The linearised version is discretised for CN, using (8.54),

$$\begin{aligned} \frac{C'_i - C_i}{\delta T} = \frac{1}{2} &\left(\alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} + \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1}\right) \\ &-2KC_i C'_i \; . \end{aligned} \tag{8.59}$$

Equation (8.59) then becomes a system like (8.11), except that the middle term on the left-hand side is different:

$$C'_{i-1} + (a_{1,i} + a_{k,i} C_i) C'_i + a_2 C'_{i+1} = b_i \tag{8.60}$$

where the new coefficient is given by

$$a_{k,i} = -4K/\alpha_1 \tag{8.61}$$

and b_i is exactly as already defined in (8.14); it does not contain a term arising from the *hcr*. The system now contains, besides the constant coefficients, concentration terms that vary from step to step. In a given program, the a-coefficients can be precomputed but the multiplication with the (known) concentrations must be performed at every step. Reduction to the didiagonal form (the first step of the Thomas algorithm), described by (8.16)–(8.28) will be modified in that (8.18) becomes

$$a'_N = a_{1,N} + a_{k,N} C_N \tag{8.62}$$

and the ith equation becomes

$$a'_i = a_{1,i} + a_{k,i} C_i - \frac{a_2}{a'_{i+1}} \ . \tag{8.63}$$

The expressions for the b' are unchanged, except for the outer value, where attention must be given to the fact that the bulk concentration itself changes (decreases) with time. Since a solution is available here (see any text on physical chemistry, for example [66]), it may as well be used. In dimensionless terms and taking the factor 2 into account, it is

$$C_{N+1}(T) = (1 + 2KT)^{-1} \ . \tag{8.64}$$

Consistent with CN custom, then, the last equation in the system is

$$C'_{N-1} + (a_{1,N} + a_{k,N} C_N)\, C'_N = b_N - a_2 C'_{N+1} \tag{8.65}$$

in which the term b_N (from (8.13)) contains the bulk value at time T (the old value), but the last term on the right-hand side contains the new value for $T + \delta T$. It is important to do this correctly, for accuracy.

The above is incorporated into the example program `BP_LIN` described in Appendix C.

An Example Case; Nonlinear

We can also choose not to linearise the nonlinear term by an approximation, in which case we do not run the (minimal) risk of adding errors to the simulation by the linearising approximation. The same example as used above (8.57) and again choosing CN as the method, the dynamic (8.58) is discretised as

$$\begin{aligned} \frac{C'_i - C_i}{\delta T} = \frac{1}{2} \big(& \alpha_1 C'_{i-1} + \alpha_2 C'_i + \alpha_3 C'_{i+1} + \alpha_1 C_{i-1} + \alpha_2 C_i + \alpha_3 C_{i+1} \\ & - 2KC_i^2 - 2KC_i'^2 \big) \end{aligned} \tag{8.66}$$

in which the term in $2KC_i'^2$ is left in its nonlinear form (along with the other nonlinear form $2KC_i^2$, but this one is a known quantity and causes no trouble). This engenders a new nonlinear system of equations

$$C'_{i-1} + a_{1,i} C'_i + a_{k,i} C_i'^2 + a_2 C'_{i+1} = b_i \tag{8.67}$$

where $a_{k,i}$ is defined differently:

$$a_{k,i} = -2K/\alpha_1 \tag{8.68}$$

and also the right-hand side, b_i, is different, now containing a chemical term involving $a_{k,i}$:

$$b_i = -C_{i-1} - a_{3,i}C_i - a_{k,i}C_i^2 - a_2 C_{i+1} \ . \tag{8.69}$$

The above (8.67), of which there are N for $i = 1 \ldots N$, form a new system, which we now rewrite in a new form. The equations are the system, for $i = 1 \ldots N$,

$$f_i(\boldsymbol{D}) = D_{i-1} + a_{1,i}D_i + a_{k,i}D_i^2 + a_2 D_{i+1} - b_i \tag{8.70}$$

where the symbol D has replaced C', but b_i still is in terms of the known present-time C row. The new D values are the approximations to C', and at the start of an iteration, are equal to the known C. We seek a solution such that all the D values are equal to the C' values, at which point all f_i are equal to zero, which they are not at the start of the step. The Newton method, which will be used, takes a number of steps, correcting the D values at each step. Before we can do this, we must pay attention to the first and last equations of the set. The first one ($i = 1$) contains the boundary value D_0. If the system is that for a Cottrell experiment, then that value is zero, so that the equation is

$$f_1(\boldsymbol{D}) = a_{1,1}D_1 + a_{k,1}D_1^2 + a_2 D_2 - b_1 \tag{8.71}$$

which is very simple. If we have derivative boundary conditions and (sensibly, with unequal intervals) use a two-point approximation for G, then D_0 can be replaced by a linear form in D_1 according to the procedures described in Chap. 6. If a multi-point derivative approximation is desired, that expression will be a linear combination of several $D_i, i = 1 \ldots n-1$, which is more complicated and is not recommended. For this example, we stay with Cottrell.

The other equation needing attention is that for $i = N$, being

$$f_N(\boldsymbol{D}) = D_{N-1} + a_{1,N}D_N + a_{k,N}D_N^2 + a_2 D_{N+1} - b_N \tag{8.72}$$

which contains the bulk value D_{N+1}. This is already known, being in this case given by (8.64) for the time $T + \delta T$. It is not part of the unknown set. Note that the last term b_N, is the expression for (8.69) for $i = N$ and contains an old bulk value. It is important in the program to distinguish between these two different bulk values.

We are now ready to implement the Newton method. The $\boldsymbol{D}$ row is an approximation to $\boldsymbol{C'}$ and we wish to correct $\boldsymbol{D}$. For details of the Newton method used on a set of nonlinear equations, see a text like Press et al. [452]. More briefly here, Taylor expansion of the system (8.66) around the current D to the corrected $\boldsymbol{D} + \boldsymbol{d}$ where $\boldsymbol{d}$ is the correction term row, produces the set of equations linear in $\boldsymbol{d}$,

$$
\begin{aligned}
f(D_1 + d_1) &= f(D_1) + (a_{1,1} + 2a_{k,1}D_1)d_1 + a_2 d_2 \\
f(D_2 + d_2) &= f(D_2) + d_1 + (a_{1,2} + 2a_{k,2}D_2)d_2 + a_2 d_3 \\
&\dots \qquad (8.73) \\
f(D_i + d_i) &= f(D_i) + d_{i-1} + (a_{1,i} + 2a_{k,i}D_i)d_i + a_2 d_{i+1} \qquad (8.74) \\
&\dots \\
f(D_N + d_N) &= f(D_N) + d_{N-1} + (a_{1,N} + 2a_{k,N}D_N)d_N
\end{aligned}
$$

where now the d_i are the unknowns, the correction terms. In vector/matrix notation, defining $\boldsymbol{D} \equiv [D_1 D_2 \dots D_N]^T$, $\boldsymbol{d} \equiv [d_1 d_2 \dots d_N]^T$ and the Jacobian

$$
\boldsymbol{J} \equiv \begin{bmatrix}
(a_{1,1} + 2a_{k,1}D_1) & a_2 & & & \\
1 & (a_{1,2} + 2a_{k,2}D_2) & a_2 & & \\
 & 1 & (a_{1,3} + 2a_{k,3}D_3) & a_2 & \\
 & \ddots & & \ddots & \\
 & & 1 & (a_{1,N} + 2a_{k,N}D_N)
\end{bmatrix} \qquad (8.75)
$$

(only nonzero elements are given), this becomes

$$
\boldsymbol{F}(\boldsymbol{D} + \boldsymbol{d}) = \boldsymbol{F}(\boldsymbol{D}) + \boldsymbol{J} \cdot \boldsymbol{d} \qquad (8.76)
$$

and expecting convergence and thus setting setting $\boldsymbol{F}(\boldsymbol{D} + \boldsymbol{d})$ to zero, we get the new, now linear system,

$$
\boldsymbol{J} \cdot \boldsymbol{d} = -\boldsymbol{F}(\boldsymbol{D}) \qquad (8.77)
$$

which is a tridiagonal system that can be solved by the Thomas algorithm as usual, for $\boldsymbol{d}$. One must then either check the residual (8.76); its norm should be below some value one sets, such as 10^{-6}. Alternatively, one can check the correction vector $\boldsymbol{d}$. If its norm is below that small value, then no further iterations need be carried out. The first method requires an extra calculation, while the second always requires a last extra iteration because even if the very first iteration yields the correct set of $\boldsymbol{d}$ values, that set itself will not be zero, but the second set will be.

The above is implemented in the program `BP_NONLIN` (Appendix C). One finds that 2–3 iterations tend to be enough, and the results are very slightly better, for a given set of simulation parameters, than those from the linearised version, `BP_LIN`.

8.6.2 Coupled Equations

Coupled equations are those in which some or all of the dynamic equations have terms in more than one of the variables (concentrations). This leads, upon discretisation, to systems of discrete equations that cannot usually be solved using the Thomas algorithm because, no matter how one orders the concentration vectors, the systems correspond to matrix equations that are

more than tridiagonal or banded. It is not very long ago that this too was regarded as too hard, and the explicit method was used, as this presents no special problems – except for accuracy and computer time. Two techniques solved the problem: what will be called the **Rudolph method**, and direct matrix equation solving. The Rudolph method, as will be seen below, is itself a method of solving the matrix equations but, by proper vector ordering and blocking, makes the matrix into a block-tridiagonal system, which can be solved by a kind of block-Thomas algorithm. The technique was in fact known outside electrochemistry [280], and within, Newman having devised this for a set of coupled second-order *ode*s in 1968 [412]; this was remarked upon by White in 1978 [570] and apparently forgotten again, until reinvented by Rudolph in 1991 [476]. From then on, the method has been in use by electrochemists. Other methods exist to deal with the problem of banded matrices. The most notable of these are the "strongly implicit procedure" (SIP) of Stone [527], used recently by Alden et al. [38, 39, 42], the Krylov method, used by the same team [41, 42, 43] and by Bard et al. [72] and Welford et al. [568]; and the multigrid method [569], also used by Alden et al. [42]. The Alden et al. team made use of ready-made commercial subroutines. These techniques are not trivial to apply, and only the Rudolph method will be described here.

To illustrate the Rudolph method, we take the relatively simple two-species catalytic or EC′ mechanism

$$\begin{aligned} O + e^- &\Leftrightarrow R \\ R &\rightarrow O \end{aligned} \tag{8.78}$$

as already mentioned in Chap. 6, page 95. As mentioned there, this leads, through the obvious dynamic equations, to the discretised pair of equations for a given point along X with index i,

$$\begin{aligned} C'_{O,i-1} + a_{1,i}C'_{O,i} \qquad\qquad + a_{k,i}C'_{R,i} + a_2 C'_{O,i+1} &= b_{O,i} \\ C'_{R,i-1} \qquad + (a_{1,i} - a_{k,i})C'_{R,i} + a_2 C'_{R,i+1} &= b_{R,i} \end{aligned} \tag{8.79}$$

(we now assume that the coefficients are the same for both species and, again, that a_2 is constant). The coefficients depend on the simulation method used (CN, Laasonen or its variants) and on the sample point distribution, as already described.

This pair of equations produces, when written for all i, a system of $2N$ equations. It is convenient to order the unknown concentrations in the sequence $C_{O,0}, C_{R,0}, C_{O,1}, C_{R,1}, \ldots, C_{O,N}, C_{R,N}$ – that is, with C_O and C_R alternating. The system then becomes a pentadiagonal matrix equation, with perhaps some elements off the five diagonals, depending upon the way derivative boundary conditions are discretised. It is true that there are pentadiagonal solvers similar to the Thomas algorithm, but we stay with the Rudolph method [476] here because the example is a simple one, and the method can

be extended to more than two species, which widen the band in the matrix to more than pentadiagonal. Incidentally, in this specific case, there is a method of using something like the Thomas algorithm, using a double recursive method, but it is specific to this example, has no general utility and will not be gone into here (unpublished work by the present author).

The concentration vector is now lumped into pairs, making it into a vector of two-element vectors; let

$$\boldsymbol{C}_i \equiv \begin{bmatrix} C_{O,i} \\ C_{R,i} \end{bmatrix} \tag{8.80}$$

and this allows us to write (8.79) in the more compact form, already seen in Chap. 6, (6.57),

$$\boldsymbol{C}'_{i-1} + \boldsymbol{A}_i \boldsymbol{C}'_i + \boldsymbol{a}_2 \boldsymbol{C}'_{i+1} = \boldsymbol{B}_i \tag{8.81}$$

with the definitions

$$\boldsymbol{A}_i \equiv \begin{bmatrix} a_{1,i} & a_{k,i} \\ 0 & (a_{1,i} - a_{k,i}) \end{bmatrix} , \tag{8.82}$$

$$\boldsymbol{B}_i \equiv \begin{bmatrix} b_{O,i} \\ b_{R,i} \end{bmatrix} . \tag{8.83}$$

The last equation of this new system, for $i = N$, can now be used in the same manner as the last equation in the scalar system (8.11), as the last term on the left-hand side is known, $\boldsymbol{C}'_{i+1}$ being the bulk values. In the scalar system, the equations, each with three unknowns, were reduced to a new set, each with two unknowns, generating a new set of scalar coefficients a'_i and b'_i. The same process can be used here, but working with vectors and matrices. Applying the same approach as the above, (and therefore not needing a lot of explanation), we write, analogous to (8.18),

$$\boldsymbol{A}'_N = \boldsymbol{A}_N \tag{8.84}$$

and as in (8.19)

$$\boldsymbol{B}'_N = \boldsymbol{B}_N - a_2 \boldsymbol{C}'_{N+1} . \tag{8.85}$$

Then, analogous to the scalar process above, (8.26) and (8.27), we have the recursive relations, going backwards from N,

$$\boldsymbol{A}'_i = \boldsymbol{A}_i - a_2 (\boldsymbol{A}'_{i+1})^{-1} \tag{8.86}$$

and

$$\boldsymbol{B}'_i = \boldsymbol{B}_i - a_2 (\boldsymbol{A}'_{i+1})^{-1} \boldsymbol{B}'_{i+1} . \tag{8.87}$$

This is continued down to $i = 1$, giving the system (6.62), described on page 96.

It turns out in practice that only the inverses of the new coefficient $\boldsymbol{A}'$-matrices are needed for the last step, so only these inverses need be stored. Since the matrices are small, the inversions can be efficiently computed.

Before the last sweep can be carried out, the boundary concentration vector $\boldsymbol{C}'_0$ is needed, and how this is calculated, is fully described in Chap. 6, Sect. 6.4, starting on page 96. When this has been done, all the new concentrations can at last be computed, from the forward-sweeping recursive expressions

$$\boldsymbol{C}'_i = (\boldsymbol{A}'_i)^{-1} \left(\boldsymbol{B}'_i - \boldsymbol{C}'_{i-1} \right) \ , \tag{8.88}$$

seen to be analogous with (8.30). If the concentrations are stored separately for each species, then it might be most convenient to put each $\boldsymbol{C}_i$ vector away into its place in those arrays as soon as they are computed; it is matter of personal strategy, how to store the values.

The program `CVRUCAT` (see Appendix C) is an example of a simulation of this system, for cyclic voltammetry.

9 Other Methods

In previous chapters, those methods that are regarded as most advisable in some sense, are presented in some detail. The explicit method cannot really be said to be advisable, but it does serve as an introduction to simulation, from which one can advance to the slightly harder implicit methods mentioned in Chap. 8. In the present chapter, a large number of alternative schemes that have been advocated in the last several decades, are at least mentioned, some in more detail than others, according to the present author's estimation of the feasibility of the methods' use, or the ability of the average electrochemist to program them. This is inevitably a subjective judgement and there will be some disagreement. References are provided for the reader who wants to delve more deeply.

9.1 The Box Method

The Feldberg approach to digital simulation [229] uses a somewhat different method of discretisation, and the method is alive and well; it is, for example, the basis for the commercial program DigiSim [482]. It begins with Fick's first diffusion equation, using fluxes between boxes or finite volumes, rather than concentrations at points in the discretisation process (see below).

Rather than, as is done in this book, sampling concentration along the x-axis at a number of points, Feldberg thought in terms of boxes along the axis. Initially, the boxes were of equal length but Feldberg proposed in 1981 that boxes of unequal length were better, and suggested exponentially expanding box-lengths [231]. A few such boxes are shown in Fig. 9.1. We have selected three boxes, consecutively numbered, as shown. The middle box is indexed with i, and is bounded by the positions x_{i-1} and x_i. Its length is h_i. The formula for the expansion can be expressed as follows. We start with a first box of length h_1, chosen suitably (with perhaps a homogeneous chemical reaction in mind, so that h_1 lies well within the reaction layer). Then, each successive box has a length a fixed multiple > 1 of the length of the one before it. This is in fact precisely the same as is done in the exponentially spaced point positions, described in Chap. 7. There, the symbol γ is given to the expansion factor. Tradition has it, in the box method approach, to use a different symbol and definition for the expansion factor; our γ is equivalent to

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 145–187 (2005)
www.springerlink.com

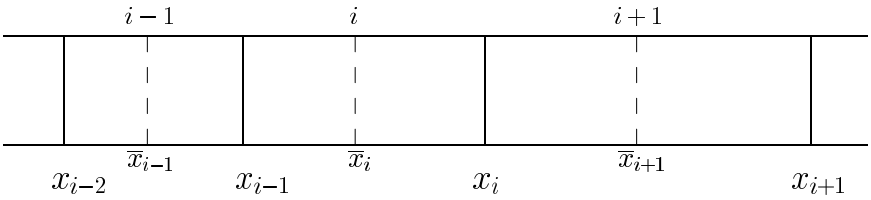


Fig. 9.1. Discrete boxes along x

$\exp(\beta)$ in the box terminology. We can thus directly describe the outer box boundaries in the same terms as in (7.19), and arrive at Feldberg's equation,

$$x_i = h_1 \frac{\exp(i\beta) - 1}{\exp(\beta) - 1} \; . \tag{9.1}$$

This is the formula in [231], seen again in Rudolph's chapter in [477]. Also, the box lengths themselves are given by

$$h_i = h_1 \exp((i-1)\beta) \; . \tag{9.2}$$

This expanding box strategy is mathematically equivalent to the transformation from X into Y as described for point positions in Chap. 7, (7.3), as is shown in Appendix B. Its implementation in the discretisation process is however different.

The way this is used is as follows. Fick's first law is used, and fluxes into and out of box i are considered. For this, we need to assign distances between successive boxes, and here a small difficulty arises. For boxes of equal length, the distance is simply that length, stretching from box midpoint to the next box midpoint. With boxes of unequal lengths, this leads to inaccuracies. What is done instead is to (mentally) map the position x onto an index space, the i's in Fig. 9.1. These have equal intervals of size unity. The assigned midpoint of a given box indexed with i is then at $i - \frac{1}{2}$, and this transforms to the midpoint positions marked in Fig. 9.1 as $\overline{x}_{i-1} \ldots \overline{x}_{i+1}$, at the dashed lines. Distances between boxes are then the distances between these points. The points are given by

$$\overline{x}_i = h_1 \frac{\exp((i - \frac{1}{2})\beta) - 1}{\exp(\beta) - 1} \; . \tag{9.3}$$

Now, the flux f_1 going into box i is

$$f_1 = -AD \frac{c_i - c_{i-1}}{\overline{x}_i - \overline{x}_{i-1}} \tag{9.4}$$

where A is the cross-sectional area of the box and D is the diffusion coefficient. The flux f_2 going out of the ith box is

$$f_2 = -AD \frac{c_{i+1} - c_i}{\overline{x}_{i+1} - \overline{x}_i} \; . \tag{9.5}$$

The resultant flux into box i is the difference between the two,

$$f = f_1 - f_2 \tag{9.6}$$

which has units of moles per second. We want concentration changes, so we must multiply by the time interval to get moles, and by the box volume to get δc_i. The time interval is δt and the box volume V_i is

$$V_i = Ah_i \ . \tag{9.7}$$

All this leads to the equation for the change in c_i,

$$\delta c_i = \frac{D\delta t}{h_i}\left(\frac{c_{i+1} - c_i}{\overline{x}_{i+1} - \overline{x}_i} - \frac{c_i - c_{i-1}}{\overline{x}_i - \overline{x}_{i-1}}\right) \tag{9.8}$$

which, using Feldberg's [231] and Rudolph's [477] notation, is now expressed in the form

$$\delta c_i = D^*_{2i}(c_{i+1} - c_i) - D^*_{1i}(c_i - c_{i-1}) \ . \tag{9.9}$$

The D^* coefficients can be worked out from (9.8), substituting for the $\overline{x}$ terms using (9.3). A minor problem may be encountered in simplifying the denominators in the two terms in brackets on the right-hand side of (9.8). As an example, consider the first of these. It is simplified in the following manner. From (9.3),

$$\overline{x}_{i+1} - \overline{x}_i = h_1 \frac{\exp((i + \frac{1}{2})\beta) - \exp((i - \frac{1}{2})\beta)}{\exp(\beta) - 1} \tag{9.10}$$

and dividing top and bottom of this fraction by $\exp((i - \frac{1}{2})\beta)$, we are left with

$$\overline{x}_{i+1} - \overline{x}_i = h_1 \exp((i - \tfrac{1}{2})\beta) \ . \tag{9.11}$$

The second denominator term can be simplified in an analogous manner, dividing by $\exp((i - \frac{3}{2})\beta)$. This leads to the expressions, for $i > 1$,

$$\begin{aligned} D^*_{2i} &= D^* \exp(2\beta(\tfrac{3}{4} - i)) \\ D^*_{1i} &= D^* \exp(2\beta(\tfrac{5}{4} - i)) \end{aligned} \tag{9.12}$$

with

$$D^* = \frac{D\delta t}{h_1^2} \ . \tag{9.13}$$

For the very first box ($i = 1$) there is another small problem. There is no box at $i = 0$, where

$$x_1 = h_1 \tag{9.14}$$

and from (9.3) ($i = 1$),

$$\overline{x}_1 = h_1 \frac{\exp(\frac{1}{2}\beta) - 1}{\exp(\beta) - 1} \ . \tag{9.15}$$

Inserting this appropriately into (9.8) results in the two coefficients

$$D_{21}^* = D^* \exp(-\tfrac{1}{2}\beta)$$
$$D_{11}^* = D^* \frac{\exp(\beta) - 1}{\exp(\tfrac{1}{2}\beta) - 1} \,. \tag{9.16}$$

The above equations are all given in [231] and [477].

Consideration of (9.9) reveals that it is of the same form as that shown for the point method using arbitrarily spaced points, (8.8) in Chap. 8. We can proceed from here in the same way as in that chapter. That is, all methods described there (or even the explicit method) can be applied. By dividing (9.9) by δt, we can even go into a MOL-type method (see below). There is thus no need to describe the procedure further from here.

As a final word on the box method, it should be mentioned that in his recent publications on unequal intervals, Rudolph [478, 479, 481] makes a strong case for the box method. His initial aim in these papers was to show that discretisation in the point method, on equal intervals in the transformed space (as described in Chap. 7) is not as accurate as had been supposed. Rudolph devised an improved way of discretising the transformed diffusion equation [479, 480] (the same as the present (7.11) in Chap. 7, derived by Bieniasz [108]), and states that the box method with exponentially expanding intervals, as described above, is as accurate as when using this improved formula. It seems that the use of fluxes is the cause of the accuracy of the box method, even though computed concentration values might be less accurate. Rudolph refers [481] to exponential convergence of calculated flux values, using this method. This is supported by the literature on the control volume method [435]. Patankar [435] writes that "even coarse-grid solution exhibits *exact* integral balances". The technique has been used in two other electrochemical works [67, 323]. There is as yet no agreement on this, and further study is needed. Certainly Rudolph's 2004 paper [481], showing very rapid exponential convergence of the computed flux, makes a strong case for this method.

9.2 Improvements on Standard Methods

Both the explicit and implicit methods already described have been improved to greater accuracy and, hopefully, greater efficiency.

9.2.1 The Kimble and White Method

Kimble and White [338] developed a scheme which, as described and intended, was somewhat awkward to use and limited the possible number of points in time and space. The method is mentioned in other chapters for its

use as a high-order start for BDF (for which it did indeed work, but not with great efficiency). It is perhaps best described in two stages. Consider Fig. 9.2, a modest-sized grid on which the KW method is to be used, representing positions in time (indices j) and space (indices i). The thick bottom line represents initial conditions; the dotted line at the left is that for the boundary C_0 values, that at the right the bulk values. The vertical line at $i = 8$ lies at X_{lim}. In general, let there be $N+2$ mesh points in the horizontal, X-direction and $M+1$ in the vertical, T-direction, that is, $N \times M$ points to be calculated aside from boundary points.

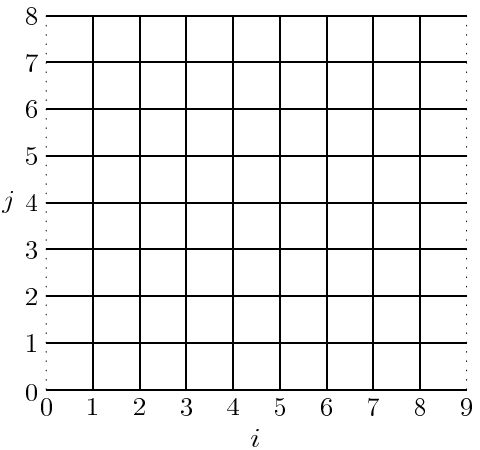
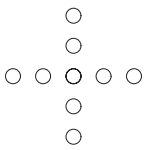


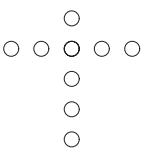
Fig. 9.2. An example grid for the KW method

Nguygen and White [414] used such a grid, albeit for the purpose of solving an elliptic problem, not involving time, so that the vertical axis was along y, the other spatial dimension. The method therefore involved second spatial differences in both directions, and they used three-point discretisations. What made their approach special is that, rather than writing one large system of equations for all $N \times M$ unknowns, which leads to a banded system, they wrote a system of matrix equations, each unknown being the whole horizontal vector. This gave them a block tridiagonal system, solvable by the available routine `BANDJ` by Newman [413], which is a precursor to the Rudolph method.

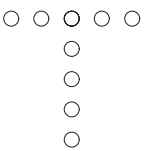
This early paper was followed by another one in 1990 by Kimble and White [338], now applying the method to a diffusion problem, and using 5-point approximations in both directions. As before, the problem was cast into a block-matrix, but because of the 5 points used for the discretisations, this was block-pentadiagonal. For most node points in the figure, the 5-point approximations yield the following computational molecule or stencil.



All points have been drawn empty, indicating what is special about the KW method. If the stencil had been drawn, as one might expect, with all points filled (known) except the top one, this would indicate that the method marches forward in time, using an explicit central differences form. Such central difference forms are all known to be unstable. The classic one is the 3-point leap-frog scheme of Richardson [468], which appears attractive intuitively, being second-order in time, but was proved unconditionally unstable in 1950 [424]. The same holds for central difference schemes using a larger number of points, as here [302]. Neither does it make a difference simply to put all the equations into one large system; the instability still appears. What made the difference here, as shown in [141], is the device the authors employed at the top of the grid. The 5-point temporal discretisation can only be used up to line $M-2$ (the top one being at index M). For index $M-1$, an asymmetric backward form is needed, using points at indices $M-4 \ldots M$ (form $y'_4(5)$ in Table A.1, Appendix A):



and for the top line at index M, a BDF form, $y'_5(5)$ in the table, was used. It is this "cap" on the whole system that stabilises it [141].



Similarly, asymmetric forward forms are used at the bottom end.

Kimble and White were aware that leapfrog methods are unstable and simply remark that this did not seem to apply to their method. Also, they mention the use of 5 points for all approximations but their table of discretisations shows that they used 6 points at the edges for the spatial second derivative. This is no doubt because, as Collatz already mentions in 1960 [170], the asymmetric 5-point second derivative is only third-order, while a 6-point formula is fourth-order, like the symmetrical 5-point ones used in the bulk of the grid. So, for the second spatial derivative at index $i = 1$, the form $y''_2(6)$ was used, and the reverse, form $y''_5(6)$ at $i = N$.

All this leads to a block-pentadiagonal system of equations in the unknown vectors representing the horizontal lines in the figure. Results appeared to be good [338] but clearly, the drawback of the method is that, for any reasonably sized grid, the system becomes large and programming is not trivial. Probably for this reason, the method has not taken on. It might, however, have application in the *ode* field, where the computational molecule reduces, as it were, to a single vertical column.

Another potential use for the KW method is as a high-order start for the BDF method, as indeed suggested by Feldberg and Goldstein [236], who dubbed this the "hyperimplicit" approach. As described in previous chapters 4 and 8, BDF has the problem of requiring starting values, if the higher-order BDF variants are to be used. If the grid in Fig. 9.2 is reduced to k time levels for a k-point BDF variant, and solved for the $k-1$ unknown levels, high-order starting values result. This process is described in Chap. 8, Sect. 8.5.2.

9.2.2 Multi-Point Second Spatial Derivatives

When using methods such as extrapolation or BDF, which are capable of high order results with respect to the time intervals, one finds that, going to orders higher than $O(\delta T^2)$ does not lead to improvements, certainly not to greater efficiency – rather the reverse, because more computing is done to achieve similar accuracies. The reason is that the error is a sum of terms involving δT^p (p being the particular method's order with respect to δT) and H^2 (for equal intervals), H being the spatial interval. This comes from the three-point spatial second derivative usually used. This term soon dominates and renders high-order time schemes useless. Thus, higher-order second spatial derivatives might help and have been studied for equal intervals [150, 152, 154, 531] as well as for unequal intervals [153]. Inspiration for this came from the KW method, described above. As with that work, 6-point asymmetric discretisation becomes desirable at points next to the boundaries, in order for all discretisations to be fourth-order with respect to the spatial interval H. The equations then are, in semidiscretised form (that is, leaving the left-hand side of the diffusion equation untouched for the moment)

$$
\begin{aligned}
\frac{dC_1}{dT} &= \frac{1}{12H^2}\left(10C_0 - 15C_1 - 4C_2 + 14C_3 - 6C_4 + C_5\right) \\
\frac{dC_2}{dT} &= \frac{1}{12H^2}\left(-C_0 + 16C_1 - 30C_2 + 16C_3 - C_4\right) \\
&\cdots \\
\frac{dC_i}{dT} &= \frac{1}{12H^2}\left(-C_{i-2} + 16C_{i-1} - 30C_i + 16C_{i+1} - C_{i+2}\right) \\
&\cdots \\
\frac{dC_{N-1}}{dT} &= \frac{1}{12H^2}\left(-C_{N-3} + 16C_{N-2} - 30C_{N-1} + 16C_N - C_{N+1}\right) \\
\frac{dC_N}{dT} &= \frac{1}{12H^2}\left(C_{N-4} - 6C_{N-3} + 14C_{N-2} - 4C_{N-1} - 15C_N + 10C_{N+1}\right)
\end{aligned}
\tag{9.17}
$$

where the first line (index 1) uses the $y_2''(6)$ form in Table A.2, and that at index N the symmetrically opposite form $y_5''(6)$. All other equations, indices $2 \ldots N-1$, use the symmetrical 5-point form $y_3''(5)$. There is, however, something special about lines 2 and $N-1$, in that, like the first and last lines, they include terms in boundary values. In practice, the C_0 values would be substituted by suitable expressions involving unknown concentration terms $C_1, \ldots,$ according to what the boundary conditions are (Chap. 6). Also, the system 9.17, being pentadiagonal, requires something more complicated than the Thomas algorithm, and one has been described [152], based on such texts as that of Engeln-Müllges and Uhlig [220] and Fletcher [250]. It involves several sweeps and, depending on the boundary condition expressions, possibly some preliminary eliminations to reduce the matrix to pentadiagonal form, if these expressions produce some extra-long equations (typically, the first and second).

This was examined in a series of works, using BDF [152], extrapolation [531], RK (see below) [150] and BDF with the KW start [154]. The latter did produce highly accurate starting vectors but due to the high computational overhead, was found less efficient than some less accurate BDF starts. Overall, the most efficient methods were those employing fourth-order extrapolation, followed closely (and surprisingly) by the simple BDF start with the time correction [154], mentioned in Sect. 4.8.1 on page 58, and Sect. 8.5.2, page 131.

In practice, the (6,5) approach is, at present, limited by the fact that, in the form presented here, it applies to equal intervals. A slight improvement with unequal intervals, using a 4-point spatial second derivative, is described in Chap. 8, and this might be sufficient improvement, at little cost in terms of desk work [143]. It has been applied to the ultramicroelectrode [532], see Chap. 12.

9.2.3 DuFort-Frankel

The (*ode*-) method called leapfrog has been mentioned in Chap. 4, where (4.38) describes it. This was used by Richardson [468] to solve a parabolic *pde*, apparently with success. The computational molecule corresponding to this method is

In this scheme, the temporal derivative is formed by the central (second-order!) difference between the upper and lower points, the second spatial derivative being approximated as usual. This makes the discretisation at the index i in space,

$$\frac{C_i' - {}'C_i}{2\delta T} = \frac{1}{H^2}\left(C_{i-1} - 2C_i + C_{i+1}\right) \tag{9.18}$$

(adhering to the notation used for BDF as in Chap. 8, page 132, where ${}'C_i$ denotes C_i at time $T - \delta T$). The scheme is clearly explicit.

Leapfrog is used with apparent success to solve hyperbolic *pde*s [528], but was proved unconditionally unstable for parabolic *pde*s in 1950 [424]. Richardson had been lucky, in that the instabilities had not made themselves felt in his (pencil and paper) calculations, in the course of the few iterations he worked.

DuFort and Frankel [216] devised a modification to this scheme in 1953 that stabilises it:

The time derivative is still a central difference but the spatial second derivative now leaves out the central point, substituting for it the mean of the past and future points. Thus, the discretisation is

$$\frac{C'_i - {}'C_i}{2\delta T} = \frac{1}{H^2}\left(C_{i-1} - {}'C_i - C'_i + C_{i+1}\right) \tag{9.19}$$

which is still explicit for C'_i when rearranged but known to be stable for all $\delta T/H^2$ [350]. This formula received some attention among electrochemists for some time [124, 233, 361], some [233] calling it "FQEFD" (fast quasi-explicit finite difference). It shares with BDF (or FIRM) the problem of start-up, since at the first step, a row of values at $T = -\delta T$ are needed. This was mentioned by Marques da Silva et al. [382], who studied this scheme, along with that of Saul'yev (see below) and its offshoots, and hopscotch (see also below). They also mention another problem with DuFort-Frankel, shared with hopscotch, as pointed out by Feldberg [232]. Both DuFort-Frankel and hopscotch, being stable for any λ, invite the use of large λ or δT values. This should result in a fast propagation of changes in the concentration profile from what is happening at the electrode. However, because these schemes are explicit, generating one new value at a time only from old values, such changes can only advance into the cell's interior space one interval at a time. This compares with implicit methods for which the whole profile is always calculated together at each step. For this reason, Feldberg [232] writes of the "propagational inadequacy" of hopscotch, and, in a private communication with Marques da Silva et al., also of the DuFort-Frankel scheme [382]. Both schemes thus perform less and less well at large δT, nullifying the advantages that might have come from the unconditional stability. This problem had been pointed out in the numerical text of Carnahan et al. [161, p. 440, Fig. 7.5] for the explicit method itself. Also, most textbooks mention the inconsistency problem of DuFort-Frankel in the case when $\delta T/H \gg 0$; in fact, the authors themselves mention this in their 1953 paper [216]. More about this can be read in Chap. 14.

The DuFort-Frankel scheme has apparently been dropped in favour of more interesting schemes such as BDF, which can be driven to higher orders, and for which the start-up problem has been overcome (Chap. 4).

9.2.4 Saul'yev

A perhaps more interesting method is that of Saul'yev [496] (and apparently independently, the same idea, of Barakat [70] a short time later). The method is explicit, which makes programming easier than implicit methods, and is capable of improvements over the original idea. There are two basic variants that make up the building blocks for improvements. The **LR** variant, as the name implies moves from left (that is, from $X = 0$) to right (higher X), generating new values at the next time level. The computational molecule for this is

and the diffusion equation is discretised from the four points in the form

$$\frac{C'_i - C_i}{\delta T} = \frac{1}{H^2}\left(C'_{i-1} - C'_i - C_i + C_{i+1}\right) \tag{9.20}$$

which is seen to be a sort of tilted second derivative on the right-hand side. The left-right progression is explicitly possible because the left-most element has already been computed in the previous step. The equation is rearranged into a form explicit for C'_i. Obviously, this leaves the problem of how to start, for which the boundary value C'_0 is needed. This will be described below. The above equation can be expressed in the form, explicit for C'_i,

$$C'_i = a_1(C'_{i-1} + C_{i+1}) + a_2 C_i \tag{9.21}$$

with the constants defined as

$$a_1 = \frac{\lambda}{1+\lambda} \tag{9.22}$$

and

$$a_2 = \frac{1-\lambda}{1+\lambda}\,. \tag{9.23}$$

The other variant is **RL**, moving from right to left:

and the discretisation is

$$\frac{C'_i - C_i}{\delta T} = \frac{1}{H^2}\left(C_{i-1} - C_i - C'_i + C'_{i+1}\right)\,. \tag{9.24}$$

When rearranged so as to be explicit for C'_i, it becomes

$$C'_i = a_1(C_{i-1} + C'_{i+1}) + a_2 C_i \tag{9.25}$$

with the same definitions of the constants. The only difference is in the superscripts of the first term on the right hand side, and in the order of evaluation, here from right to left.

It remains to describe how to handle the boundary value C'_0. Clearly, for the RL variant, there is no problem because the last concentration value calculated is C'_1, and C'_0 can then be computed from all the other C' values, now known, according to the boundary condition. This leaves the LR problem. If the boundary concentration is determined as such (the Dirichlet condition, for example the Cottrell experiment), then this is simply applied. It is with derivative (Neumann) boundary conditions that there is a (small) problem. Here, we know an expression for the gradient G at the electrode. For simplicity, assume a two-point gradient approximation at time $t + \delta T$ (G'),

$$G' = \frac{C'_1 - C'_0}{H} \tag{9.26}$$

and this can be coupled with the first LR expression for C'_1 from (9.21), setting $i = 1$, and the two equations solved for C'_0 (and C'_1). The LR process can then begin. If more points are to be used for the gradient approximation, then more LR expressions must also be added, and a correspondingly larger system of equations needs to be solved. This has been described [144] in some detail.

This is as much as will be said here about the mechanics of the Saul'yev method; the reader can take it from here, as it is quite simple. Some further remarks are however in order.

Both the LR and RL variants, despite being explicit, are said to be stable for all λ values, which is a great advantage. Also, the method does not share with DuFort-Frankel and hopscotch the propagational inadequacy problem [232] mentioned above because both variants amount to a recursive algorithm, each newly calculated element carrying with it some component from all previously calculated elements.

There are drawbacks, however. It is clear from the above computational molecules, that the second, spatial derivative is approximated in an asymmetric manner, and although these approximations are in fact second-order with respect to the interval H, they are not as good as, say, the Crank-Nicolson ones. Both LR and RL, taken by themselves, do not produce very good results. It was not long after Saul'yev's book in 1964, that Larkin (in the same year) published some extensions, as did other workers [223, 367, 368]. The asymmetry of each of the two variants suggests combining them in some manner. Larkin [352] listed four strategies:

1. use the LR variant only;
2. use the RL variant only;

3. use the LR and RL variants alternately at each iteration;
4. use the LR and RL variants independently at each iteration and average the result.

Liu [367, 368] later added a modification, using one extra point at the bottom advancing end of the molecules shown above, and showed that this made the schemes more accurate and that they were still stable. Evans and Abdullah [223] developed what they called group explicit methods (GEM) based on Saul'yev, in which the LR and RL schemes were combined in larger computational molecules.

Electrochemists first investigated the Saul'yev method in 1988 and 1989 [381,382], including GEM, and the incorporation of implicit boundary values was added later [144]. The result of these studies is broadly that the last of Larkin's options above, averaging LR and RL, is the best. This has about the same accuracy as Crank-Nicolson, and could be considered to be easier to program. The third option, alternating LR with RL, produces oscillations.

The stability of the Saul'yev schemes in the electrochemical context with mixed boundary conditions, was examined [112, 118]. Surprisingly, it was found that the LR variant can be unstable with mixed boundary conditions. There exists, for any number N of intervals in space, a maximum λ value in the discrete equation, above which the LR scheme becomes unstable. Fortunately, it is rather difficult to attain this condition in practice. Since these studies, Deng [207] has used the various Saul'yev schemes and offshoots, and cites several Chinese studies also using Saul'yev variants, but they have found little application elsewhere.

9.2.5 Hopscotch

In 1965, Gordon [272] reported some studies of what he called nonsymmetric difference equations, meaning schemes like that of Saul'yev and the Peaceman-Rachford ADI scheme (see Chap. 12), in which not all points are treated alike. One of his new ideas was what he called the "explicit-implicit" scheme. It is as follows (using the simple example of a 1-D simulation). As usual, we move along time with index j and along space (X) with index i, starting with $j = 1$, having set the initial values for $j = 0$. The X points are indexed from zero to $N+1$, with $X_0 = 0$ and X_{N+1} lying in the bulk, outside the diffusion space.

If j is even, then we first explicitly compute new points for all odd i, that is,

$$C'_i = C_i + \lambda(C_{i-1} - 2C_i + C_{i+1}) \,, \tag{9.27}$$

and then using the implicit formula (the same as backward implicit BI) on all points with even i,

$$C'_i = C_i + \lambda(C'_{i-1} - 2C'_i + C'_{i+1}) \,. \tag{9.28}$$

The interesting thing here is that in contrast with BI, the values for C'_{i-1} and C'_{i+1} are already known from the run of (9.27), lying at odd values of i, so that (9.28) can be rearranged explicitly for C'_i. At the next iteration, j will be odd, and the explicit calculation is done on all even i, followed by the implicit calculation on all points with odd i. In this way, alternating the sets being computed explicitly and implicitly, a certain symmetry is produced.

Gordon also showed that the scheme is convergent and stable for all λ. The scheme was taken up by Gourlay in 1970 [275], who tightened up the mathematical notation, and applied the scheme to 2-D numeric problems, as well as introducing the trick of overwriting values in the first (explicit) step, so that only one array of values is needed. Gourlay coined the name "hopscotch" for this method, by which it has been known since then, and usually only Gourlay is cited. There were follow-up papers [224, 276]. It also became clear that the method was closely related to others like ADI [275] (to be mentioned for 2-D).

The enthusiasm for hopscotch arose from the fact that here was a method with an accuracy thought to be almost comparable with that of Crank-Nicolson, but which was an explicit computation at every step, not requiring the solution of linear systems of equations, as other implicit methods do. It was also stable for all λ, thus making it possible to use larger time steps, for example. The convenience of the point-by-point calculation has occasionally led workers to call the method "fast" [235].

Shoup and Szabo [507, 508, 509, 510] brought the method to electrochemistry, using it to simulate diffusion at a microdisk electrode. This was a problem at the time. A proper implicit scheme leads to rather large banded systems of equations (see Chap. 12), and workers tended to use ADI, which leads to (much smaller) systems of equations. Hopscotch seemed to be the answer, as one could recalculate all points explicitly, and use large λ values (in both directions). Feldberg [235] used the method to simulate processes at a rotating ring-disk electrode, citing stability and ease of use. Other electrochemists followed [52, 251, 252, 354, 355, 394, 434, 489]. There soon appeared criticisms, however. Ruzić [484] commented that Shoup and Szabo had misrepresented the normal explicit method by stating that it required two concentration arrays, and showed how this could easily be avoided by using two scalar concentration values trailing behind the values treated, while overwriting all values as they are calculated (this is the trio of variables, `C1`, `C2`, `C3`, used in the example program `COTT_EX`, see Appendix C). Also, Ruzić showed that some simple known improvements [494] to Feldberg's explicit method improved its accuracy to something close to that of the hopscotch method, so that the latter was not needed. Shoup and Szabo had indeed shown in their 1984 paper [509], that hopscotch's accuracy declines badly at λ values exceeding unity, so the ability to use large λ cannot be cited as an advantage of the method. Ruzić's polemic was rebutted by Shoup and Szabo [511], who admitted some of the points made but then launched a discussion on the

precise implementation of the Feldberg (box-) method which, unlike the point method, allows a number of interpretations and tricks to improve the results. The improvement described by Ruzić, shown in his example program and based on Sandifer and Buck [494], amounts to the use of the point method.

In 1987, Feldberg [232] pointed out the most serious drawback of hopscotch. The problem is that, at each step forward in time, application of the two (9.27) and (9.28) can propagate a perturbation at a given point in the profile (for example, at the electrode) only by a single interval in space. If large time intervals are used, then one would expect such changes to make themselves felt over a number of neighbouring points, but hopscotch cannot do this. Feldberg writes of the "propagational inadequacy" of the hopscotch method. As mentioned above, it shares this with the DuFort-Frankel method and also with the explicit method. With the latter, however, the stability limit on λ prevents the use of time intervals large enough for this inadequacy to matter, while for hopscotch (and DuFort-Frankel) there is the possibility and temptation to use larger time intervals. At values that can be used in the explicit method, hopscotch is only marginally better than explicit and this, together with the propagational inadequacy feature, suggests that hopscotch is not a method of choice, despite the ease of programming, both for one- and two-dimensional simulations.

9.2.6 Runge-Kutta

The RK variants are described for *ode*s in Chap. 4, from page 54 and, for a system of *ode*s, from page 66. There, only the Euler method is detailed, but in terms of RK terminology, from which the higher-order variants follow easily. The description there will not be expanded here, for reasons given below. When solving a *pde* or a system of such, one way is to "semidiscretise" the equation(s), meaning that only the right-hand side is discretised, leaving the time derivative as it is. This yields a system of *ode*s, such as (4.49), and one can then proceed with that. This is called the Method of Lines (MOL). One can either treat the boundary conditions separately, or add them to the system, in which case the system becomes a **DAE** system, and requires other methods to solve for it, as briefly mentioned in Chap. 4.

RK initially attracted attention in electrochemical digital simulation because of homogeneous chemical reactions. With explicit simulations, it was realised that there was a problem if the term $K\delta T$ was of appreciable magnitude [234, 246]. Nielsen et al. [418] point out that, if this term causes more than a few percent change in a concentration, the simulation will be inaccurate. Early on it was suggested to treat the chemical term more accurately. Feldberg and Auerbach [234] used the known analytical solutions for a first- and second-order chemical reaction for the chemical term, and Flanagan and Marcoux [246] followed, suggesting RK integration for those cases in which analytical solutions are not known. The RK method was then used by Nielsen et al. [418].

It was realised then that the method of Nielsen et al. [418] had a defect, limiting its accuracy. The diffusional and chemical terms were calculated separately, in sequence. That is, first diffusional changes are applied to the concentrations, and then the chemical reaction is allowed to run, on the changed concentrations. This is the "sequential method" also used for the plain explicit (Euler) method for both terms, where it has been shown to be consistent mathematically [485]; see also Appendix B. No proof of the consistency of the method, when RK is applied to the chemical terms, is known, however. Clearly, this technique uncouples the two processes taking place, diffusion and chemical reaction. This was remedied [135] in a work where RK was applied to the whole system of equations, thus taking care of the coupled nature of the two processes. It was found [135] that using RK2, a modest efficiency gain of about a factor 3, in terms of computer time used, was achieved, compared with the plain explicit method, in order to reach a given target accuracy in model simulations. This is not very much and the method has the additional drawback of a limit on the size of λ, the same as the explicit method, 0.5. Nevertheless, this whole-system RK method has seen some use since then, notably by the Lemos school [357,358,359,360,466], who emphasise the MOL nature of their approach, Gosser [273], and Barker [79]. Accuracy contours were computed for the method, among others [147] and a stability analysis was published [94], as affected by the chemical reaction (the reaction lowers the limit on λ). In the course of an investigation of higher-order discretisations of the spatial second derivative, RK was once again tested [150] and once again found not especially promising; using 5-point discretisation, the limit on λ decreases to 0.375.

It is therefore concluded that this method, using explicit RK as described in Chap. 4, is not worthwhile mainly because of the λ limitation.

There are, however, implicit variants of RK, and these may have promise. There are several classes of these, see a thorough text on the subject [284, 286]. One of these classes, the Rosenbrock method, has been recently examined [100,113, and see references therein] and found very efficient. This is described in its own Sect. 9.4, below.

9.2.7 Hermitian Methods

Kopal [341] describes Hermitian interpolation, as used by Hermite. The essence of this is that not only function values at grid points are used, but also derivative values. For a given number of grid points used in a particular approximation formula, this results in a higher order accuracy with respect to the grid intervals. Although Hermite used this only for interpolation, the term is now used more generally, referring to the characteristics mentioned above. Three Hermitian methods have been used in electrochemical simulations, up to the time of writing, and two of them are due to Bieniasz.

Numerov/Douglas

In 1924, the Russian astronomer Numerov (transliterating his own name as Noumerov), published a paper [421] in which he described some improvements in approximations to derivatives, to help with numerical simulations of the movement of bodies in the solar system. His device has been adapted to the solution of *pde*s, and was introduced to electrochemistry by Bieniasz in 2003 [108]. The method described by Bieniasz is also called the Douglas equation in some texts such as that of Smith [514], where a rather clear description of the method is found. With the help of the Numerov method, it is possible to attain fourth order accuracy in the spatial second derivative, while using only the usual three points. The first paper by Bieniasz on this method treated equally spaced grids, and was followed by another on unequally spaced grids [107]. The method makes it practical to use higher-order time derivative approximations without the complications of, say, the (6,5)-point scheme described above, which makes the solution of the system of equations a little complicated (and computer time consuming).

The description in Smith [514, pp. 137-] is followed here. It starts with a statement that a second derivative can be approximated by

$$\frac{\partial^2 u}{\partial x^2} \approx \left(\delta_x^2 u - \tfrac{1}{12}\delta_x^4 u + \tfrac{1}{90}\delta_x^6 u - \dots\right) , \tag{9.29}$$

where the symbol $\delta_x^n u$ denotes the operation δ_x^n on u corresponding to an approximation to the nth derivative. In particular, we have

$$\delta_x^2 u = u_{i-1} - 2u_i + u_{i+1} , \tag{9.30}$$

the familiar three-point form, second order in the interval h between the (equally spaced) points at indices $i-1, i, i+1$. It does not contain the interval h. Also, if δ^2 operates on itself, it becomes δ^4, etc. It is an operator, but can in this sense be treated as a multiplier. It will be seen that we do not need to define higher derivatives than the second, operator δ^2.

Smith does not explain the origin of (9.29), but a derivation can be found in Lapidus and Pinder [350, pp. 19-], to which the reader is referred. The form seen in (9.29) is one of several equally valid forms, but is the one chosen in this context, as it allows the Numerov device.

The method will be described as applied to BI, which is the basis for both extrapolation and BDF, both of which can be driven to fourth order accuracy, which is also achieved by the Numerov device applied to the right-hand side of the diffusion equation,

$$\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial X^2} . \tag{9.31}$$

Using the usual notation, C_i' denoting the next point in time after the present value C_i, i being the index along the X axis, we now discretise the left-hand

side only, according to BI (see Chap. 8, but now assuming equal intervals), and use (9.29) for the right-hand side:

$$C'_i - C_i = \lambda \left(\delta_X^2 C'_i - \tfrac{1}{12}\delta_X^4 C'_i + \tfrac{1}{90}\delta_X^6 C'_i - \dots\right) . \tag{9.32}$$

We need not concern us with the implementation of the higher-order derivatives, as will shortly be clear. Now both sides are operated on by $(1 + \frac{1}{12}\delta^2)$, which is the same as adding to each side the operation $\frac{1}{12}\delta_X^2$ on that side. This gives

$$\begin{aligned} &C'_i - C_i + \tfrac{1}{12}\delta_X^2(C'_i - C_i) \\ &\quad = \lambda\Big(\delta_X^2 C'_i - \tfrac{1}{12}\delta_X^4 C'_i + \tfrac{1}{90}\delta_X^6 C'_i - \dots + \tfrac{1}{12}\delta_X^4 C'_i - \tfrac{1}{144}\delta_X^6 C'_i + \dots\Big) \end{aligned} \tag{9.33}$$

and it is seen that on the right-hand side, there are now only terms in δ_X^2 and δ_X^6, the δ_X^4 having cancelled out. We can safely ignore the δ_X^6 and higher terms, and now we have only δ_X^2 terms on both sides of the equation. Now expanding according to the definition of the operation δ_x^2 (9.30), we get

$$C'_i - C_i + \tfrac{1}{12}(C'_{i-1} - 2C'_i + C'_{i+1} - C_{i-1} + 2C_i - C_{i+1}) = \lambda(C'_{i-1} - 2C'_i + C'_{i+1}) \tag{9.34}$$

which, multiplying by 12 and gathering terms, becomes the familiar form seen in Chap. 8, the ith equation of system (8.11) for a general implicit method,

$$C'_{i-1} + aC'_i + C'_{i+1} = b_i , \tag{9.35}$$

where now

$$\begin{aligned} a &= \frac{10 + 24\lambda}{1 - 12\lambda} \\ b_i &= \frac{C_{i-1} + 10C_i + C_{i+1}}{1 - 12\lambda} . \end{aligned} \tag{9.36}$$

The difference is that this discretisation is $O(H^4)$. The system can be solved as easily by the Thomas algorithm as, say, the usual Laasonen or CN system, but now it will be worthwhile applying a high-order process in the time direction. The easiest one is extrapolation, described above, and it ought to be fourth order, so as to match that of the spatial second derivative. Bieniasz tested the method with three simulation algorithms, comparing with the normal, second-order discretisation: BI (no difference, because of the first-order time derivative), second-order extrapolation (not much difference, the second-order not providing a match for the fourth order) and the Rosenbrock scheme using ROWDA3, which showed a marked improvement in efficiency. Unfortunately, he did not attempt fourth-order extrapolation, which might be expected to perform about as well as Rosenbrock, and would be easier to implement, being simply a series of BI steps.

This method is worth investigating further. An analysis of the stability of the formulae resulting from the method is yet to be done. There are some features to note. Considering the constants definitions above (9.36), there is an apparent problem if $\lambda = 1/12$. In fact, if the whole (9.35) is multiplied by $1 - 12\lambda$, then this problem becomes a possible advantage, as that equation then simplifies, for $\lambda = 1/12$, to

$$12C_i' = C_{i-1} + 10C_i + C_{i+1} \; . \tag{9.37}$$

It is not clear whether this is a good formula in practice, and in any case, the λ value is inconveniently small.

Note also that, if there are homogeneous chemical reaction terms on the right-hand side of (9.31), they can be accommodated without problems; they will lead to some additional terms operated on by δ_X^2. What must not be present are convection terms, since these are spatial first derivatives, making the Numerov method, in this form, impossible to use. However, Bieniasz has devised an improved version, called the "extended Numerov method" [110], which indeed can handle first spatial derivatives and thus convective systems.

Hermitian Current Approximation

As already described in some detail in Chap. 3, a one-sided first derivative such as the current approximation G can be raised to higher-order by a Hermitian scheme, as introduced by Bieniasz [108]. This can then be used both to obtain better current approximations, and also in those cases where G enters a boundary condition. For the simpler case of the current approximation on a concentration grid already calculated, see the relevant Sect. 3.6 in Chap. 3. Here we need to go into some detail on the boundary conditions application.

There are simulation cases (for example using unequal intervals) where it is desirable to use a two-point approximation for G, both for the evaluation of a current, and as part of the boundary conditions. In that case, an improvement over the normally first-order two-point approximation is welcomed, and Hermitian formulae can achieve this. Two cases of such schemes are now described: that of controlled current and that of an irreversible reaction, as described in Chap. 6, Sect. 6.2.2, using the single-species case treated in that section, for simplicity. The reader will be able to extend the treatment to more species and other cases, perhaps with the help of Bieniasz' seminal work on this subject [108]. Both the 2(2) and 2(3) forms are given. It is assumed that we have arrived at the reduced didiagonal system (6.3) and have done the u-v calculation (here, only u_1 and v_1 are needed).

We must also specify the time integration method used, because the Hermitian scheme makes use of terms in dC/dT, which must be consistent with the time integration. We assume the three-point BDF method, second-order in time, so that an improvement in the usual two-point G-approximation to second or perhaps third-order (in space) will be appropriate.

In the cases to be described below, we have a simple F-function, containing only a term in dC/dT (see the outline in Chap. 3), which needs approximating. With BDF, this is consistently represented as

$$\frac{dC_i}{dT} = \frac{'C_i - 4C_i + 3C_i'}{2\delta T} \tag{9.38}$$

with $'C_i$ the concentration at $T - \delta T$. If the rational BDF start is used (Sect. 4.8.1), then the simulation will start with a single BI step, for which we have

$$\frac{dC_i}{dT} = \frac{C_i' - C_i}{\delta T} \ . \tag{9.39}$$

For **controlled current** G and the 2(2) form, the boundary condition becomes the corrected form (note that $\phi_0 = -1/2$ and $\phi_0 = 0$),

$$G = \frac{C_1' - C_0'}{H} - \frac{H}{2} F_0 \tag{9.40}$$

and multiplying both sides by H and expanding F_0 for the BI step, results in

$$GH = C_1' - C_0' - \frac{H^2}{2}\left(\frac{C_0' - C_0}{\delta t}\right) \ . \tag{9.41}$$

Substituting $C_1' = u_1 + v_1 C_0'$ and rearranging, we obtain the solution

$$C_0' = \frac{2\delta T(GH - u_1) - H^2 C_0}{2\delta T(v_1 - 1) - H^2} \ . \tag{9.42}$$

For the subsequent BDF steps, this is

$$GH = C_1' - C_0' - \frac{H^2}{2}\left(\frac{'C_0 - 4C_0 + 3C_0'}{2\delta t}\right) \ , \tag{9.43}$$

leading finally to

$$C_0' = \frac{4\delta T(GH - u_1) + H^2('C_0 - 4C_0)}{4\delta T(v_1 - 1) - 3H^2} \ . \tag{9.44}$$

The 2(3) form ($\phi_0 = -1/3$, $\phi_1 = -1/6$) starts with

$$G = \frac{C_1' - C_0'}{H} - \frac{H}{3} F_0 - \frac{H}{6} F_1 \ . \tag{9.45}$$

Expanding and substituting for both C_0' and C_1' (the latter arising in F_1), the final result for the BI step is

$$C_0' = \frac{6\delta T(GH - u_1) + H^2(u_1 - 2C_0 - C_1)}{6\delta T(v_1 - 1) - H^2(v_1 + 2)} \ , \tag{9.46}$$

while for the BDF steps it becomes

$$C_0' = \frac{12\delta T(GH - u_1) + H^2(3u_1 + 2'C_0 - 8C_0 + 'C_1 - 4C_1)}{12\delta T(v_1 - 1) - 3H^2(v_1 + 2)} \,. \tag{9.47}$$

For the **irreversible case** with dimensionless heterogeneous rate constant K, G is given as

$$G = KC_0' \tag{9.48}$$

and using this instead of G as above, we have for the 2(2) scheme and BI,

$$KHC_0' = C_1' - C_0' - \frac{H^2}{2}\left(\frac{C_0' - C_0}{\delta t}\right) \tag{9.49}$$

which rearranges to

$$C_0' = \frac{2\delta T u_1 + H^2 C_0}{2\delta T(KH - v_1 + 1) + H^2} \tag{9.50}$$

or for the BDF steps

$$C_0' = \frac{4\delta T u_1 - H^2('C_0 - 4C_0)}{4\delta T(KH - v_1 + 1) + 3H^2} \,. \tag{9.51}$$

For the 2(3) scheme and BI,

$$C_0' = \frac{6\delta T u_1 - H^2(u_1 - 2C_0 - C_1)}{6\delta T(KH - v_1 + 1) - H^2(v_1 + 2)} \tag{9.52}$$

and for the BDF steps,

$$C_0' = \frac{12\delta T u_1 - H^2(3u_1 + 2'C_0 - 8C_0 + 'C_1 - 4C_1)}{12\delta T(KH - v_1 + 1) - 3H^2(v_1 + 2)} \,. \tag{9.53}$$

Some experiments show that the 2(2) forms are sufficient here, the 2(3) forms not leading to further improvement in accuracy. This is no doubt because the three-point BDF algorithm used, started with a BI step, is second order accurate in time, so a third-order form cannot improve the accuracy. A higher-order algorithm, such as ROWDA3 as used by Bieniasz [108] would make the higher 2(3) form more useful.

Time-integration schemes other than BDF require other expressions of dC/dT to be consistent with the time integration scheme itself. For CN, this cannot be done consistently very well. Bieniasz showed how to do it for extrapolation and for the Rosenbrock ROWDA3 scheme [108]. The reader is referred to that paper for details, where still higher-order forms are found. The paper makes it clear that extremely small errors can be achieved by using this method.

Method of Wu and White

Wu and White [577] have described a new method that is reminiscent of the earlier work of Kimble and White [338] but makes use of the Hermitian method (that is, using derivatives) to achieve higher-order solutions for several concentration rows at a time. They also suggest, but do not demonstrate, the use of their new scheme as a possible start-up for BDF. The reader is referred to their paper for details.

9.3 Method of Lines (MOL) and Differential Algebraic Equations (DAE)

The Method of Lines or **MOL** is not so much a particular method as a way of approaching numerical solutions of *pde*s. It is described well by Hartree [295] as the "replacement of the second-order (space) derivative by a finite difference"; that is, leaving the first (time) derivative as it is, thus forming from, say, the diffusion equation a set of ordinary differential equations, to be solved in an unspecified manner. Thus, a system such as (9.17) on page 151, can be written in the general vector-matrix form

$$\frac{d\boldsymbol{C}}{dT} = \boldsymbol{AC} + \boldsymbol{s} \ , \tag{9.54}$$

where $\boldsymbol{C}$ is the concentration vector, $\boldsymbol{A}$ is the matrix of coefficients in the system and $\boldsymbol{s}$ is a vector of known quantities arising from the particular boundary conditions. From this point on, a large variety of methods for solving this system can be used. This encompasses all the methods so far described, but the term MOL nowadays implies a particular method. This consists of using a variety of computer packages to solve the set of *ode*s, usually with a high degree of autonomy with respect to time intervals and if, for example, BDF is used (as it often is with these packages), with respect to BDF order. The word "lines" comes from the fact that the solution is advanced a "line" at a time, the line stretching along the space dimension, and advancing up the time axis.

The method has a long history. The name MOL seems to have become established around 1960. Before this, various authors either used the word "line" [254] or expressions like "on certain lines" [330] or a description of the idea. In the book by Kantorovich and Krylov [330], there is a reference to a 1934 paper [329]. It is also cited by Liskovets [366] as a source paper, along with Rothe (1930) [475], who might be the first. Hartree and Womersley [296] use, in their summary, the words "approximating by use of finite intervals in one variable, and integrating exactly in the other variable". The book by Schiesser [497] is the standard work now (he calls the method NUMOL, for numerical method of lines). Electrochemical use of MOL has been sparse. Lemos and coworkers [357, 359, 360] have investigated the method, using

various solution methods; Lasia and Grégoire [353] used it in conjunction with a professional *ode* solver package, as did Zhang and Cheh [583].

MOL is intimately bound up with another method, that of using differential algebraic equation (**DAE**) sets. It can be thought of as an extension of MOL. Therefore, MOL should be described here, and it is in fact simple. In the most popular form of MOL, the diffusion equation is discretised on a grid in the spatial dimension(s) only, leaving the time derivative as it is. This results in a set of ordinary differential equations (9.54) as seen, for example, in the system (9.17) in this chapter. There, (6,5)-point approximations are used for the spatial derivative, but this is immaterial; three-point formulae such as described in Chap. 8, (8.1) on page 119 are more commonly used. In Chap. 6 the discretisation of boundary conditions is described. The idea there is that a system like (9.17) is solved in two steps by, for example, the Thomas algorithm. At the end of the first stage, boundary conditions are expressed discretely and used in the second stage. However, another approach is to add the discrete expressions for boundary conditions to the *ode* system. These expressions are algebraic equations. For example, for the Cottrell system, the expression

$$C_0 = 0 \tag{9.55}$$

might be added to the system. Or, if the experiment is that of chronopotentiometry, discretised derivative boundary conditions will be developed and added to the system, such as (6.4) or more complex discretisations. The result is always a system of equations, some of which are differential and some algebraic. This is a DAE system, and there are professional packages for their numerical solution. The standard text is that of Brenan et al. [130], in which references are to be found to existing packages such as LSODE and DASSL [441]. These can mostly be found at the `netlib@ornl.gov`. More on these packages can be found in Chap. 16.

A little detail is appropriate here, especially as this is needed for the next Section. Consider a set of *ode*s using three-point approximation in space, such as (8.1) on page 119, for simplicity. For N internal points, there will then normally be N such equations in the set. Up to this point, the method has been either to substitute for the boundary values according to the equations describing them (the boundary conditions) or, as in the case of the implicit methods described above, to perform a Thomas process going backward from the external boundary, and then to solve for the value of C_0, for example using the u, v process as described above. If Runge-Kutta is used, one begins by generating a particular $\boldsymbol{k}_i$ vector, then uses this to calculate the k_i belonging to C_0 (and possibly to C_{N+1}, if that also changes with time), going on from there. These methods in one way or another separate the treatment of boundary values from that of the internal points. However, as mentioned above, the equations describing boundary values can also be added to the equation set. They are always algebraic equations, so that the whole set is then a DAE set. As a simple example, if we simulate chronopotentiometry

using a two-point approximation for the current, and equal spatial intervals H, the DAE set corresponding to (8.1) becomes

$$\begin{aligned} 0 &= C_1 - C_0 - HG \\ \frac{dC_1}{dT} &= \frac{1}{H^2}\left(C_0 - 2C_1 + C_2\right) \\ &\ldots \\ \frac{dC_i}{dT} &= \frac{1}{H^2}\left(C_{i-1} - 2C_i + C_{i+1}\right) \\ &\ldots \\ \frac{dC_N}{dT} &= \frac{1}{H^2}\left(C_{N-1} - 2C_N + C_{N+1}\right) \\ 0 &= C_{N+1} - 1\ . \end{aligned} \tag{9.56}$$

The first equation is a description of the (controlled) current approximation, and the last equation expresses the unity value of the outer boundary value.

There are now two principal ways of handling this set. One is to decide on some discretisation of the time derivatives, rendering the *ode*s into algebraic equations, and solving the lot, for the next time step. The method chosen might be BDF, for example, which is indeed used in the DAE solver package DASSL [441]. Seen in this light, DAE sets might be considered always to be involved; when we use the u, v mechanism along with the Thomas algorithm, we are essentially solving the DAE set in an efficient manner. The other approach goes in the opposite direction, as it were. All the *ode*s in the set are left as such, and the algebraic equations are solved along with them, using an *ode* solver. One of these is Runge-Kutta but, as was mentioned above, explicit RK is not very efficient, so an implicit method suggests itself, such as Rosenbrock, described for sets of *ode*s in Chap. 4. This is dealt with in the next section.

9.4 The Rosenbrock Method

For the basics of this method, see Chap. 4. There it was mentioned that Bieniasz introduced this method to electrochemical simulation [100], preferring ROWDA3, a third-order variant that also has a smooth response. There exists a second-order variant with a smooth response, ROS2, due to Lang [347], which might be more appropriate if second-order spatial derivative approximations are to be used. Coefficients for some variants are given in Appendix A. The object here is to describe the way Rosenbrock methods are used in the present context. The Bieniasz paper [100] shows the way (but the standard symbols, as used in Chap. 4, are used here, rather than those used by Bieniasz).

The set (9.56) (or one like it, with whatever boundary conditions we might have) is written in the compact form

$$\boldsymbol{S}\frac{d\boldsymbol{C}}{dT} = \boldsymbol{F}(T, \boldsymbol{C}) \tag{9.57}$$

where $\boldsymbol{S}$ is the selection matrix. It is diagonal, and contains zeroes in those positions where the DAE set has an algebraic equation (that is, zero on the left-hand side of, say, (9.56)) and unity in those positions corresponding to the *ode*s. In a sense, the zeroes say "zero dC/dT". On the right-hand side, the function $\boldsymbol{F}(T, \boldsymbol{C})$ is a matrix-vector function expressing the whole collection of the right-hand sides of, say, the set (9.56). The variable T is included for the important generality for those cases where some variables are time-dependent. This is the case, for example, in LSV simulations, where the potential is a function of time, or cases where the electroactive substance itself undergoes a homogeneous reaction, as in the Reinert-Berg system or, of more interest in this context, the second-order Birk-Perone system, which gives rise to nonlinear equations, both described in the subsection beginning on page 20. As was seen in Chap. 4, one of the strengths of Rosenbrock is indeed its easy handling of nonlinear sets.

Applying (9.57) to the specific set 9.56 for illustration, we have

$$\boldsymbol{S} = \begin{bmatrix} 0 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & 0 \end{bmatrix} , \tag{9.58}$$

that is, $\boldsymbol{S}$ is almost an identity matrix except that the first and last diagonal elements are zero, indicating algebraic equations. The function $\boldsymbol{F}(T, \boldsymbol{C})$ can be broken up into

$$\boldsymbol{F}(T, \boldsymbol{C}) = \boldsymbol{J}\boldsymbol{C} + \boldsymbol{s} , \tag{9.59}$$

in which the matrix $\boldsymbol{J}$ multiplies the concentration vector; it is written as $\boldsymbol{J}$ here because it is the Jacobian, and will later figure as such. Vector $\boldsymbol{s}$ arises from the constant terms of the DAE set. The example set (9.56) resolves into

$$\boldsymbol{J} = \frac{1}{H^2}\begin{bmatrix} -H^2 & H^2 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & & & & \\ & & & 1 & -2 & 1 \\ & & & & & H^2 \end{bmatrix} , \tag{9.60}$$

$$\boldsymbol{s} = \frac{1}{H^2}\begin{bmatrix} -H^3 G \\ 0 \\ \vdots \\ 0 \\ -H^2 \end{bmatrix} \tag{9.61}$$

and the $\boldsymbol{C}$ vector indexed from 0 to $N+1$.

We are now ready to invoke the Rosenbrock method. A number s of $\boldsymbol{k}_i$ vectors must be computed, s being the order chosen. The general equation for each one is an extension of that given for a pure *ode* set on page 70, (4.70), to the present DAE case, introducing the selection matrix $\boldsymbol{S}$ and following Bieniasz [100] (though with the more common notation):

$$\begin{aligned}[\boldsymbol{S} - \gamma\delta T \boldsymbol{F}_C(T, \boldsymbol{C})]\, \boldsymbol{k}_i = \gamma \Bigg(& \delta T \boldsymbol{F}\left(T + \alpha_i \delta T,\, \boldsymbol{C} + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j\right) \\ & + \boldsymbol{S} \sum_{j=1}^{i-1} c_{ij} \boldsymbol{k}_j + \gamma_i \delta T^2 \boldsymbol{F}_T \Bigg) \,.\end{aligned} \tag{9.62}$$

Here, there appear the Jacobian $\boldsymbol{F}_C$, which is in fact $\boldsymbol{J}$ as defined above in (9.60), the function $\boldsymbol{F}$ itself, applying at partly augmented T and $\boldsymbol{C}$ values, and, in case of time-dependent systems, the time derivative $\boldsymbol{F}_T$, written in short form, as it is applied to the present T and $\boldsymbol{C}$. This last term is often zero, if the system does not include functions of time.

Although (9.62) may look formidable, there are some conveniences. First of all, for linear systems, the first matrix term on the left-hand side is a constant and can be evaluated once and for all. We write

$$\boldsymbol{M} = \boldsymbol{S} - \gamma\delta T \boldsymbol{F}_C \,. \tag{9.63}$$

In fact, in practice, some further tidying up is possible, by combining the quantities δT and $\dfrac{1}{H^2}$ into the familiar λ and dividing throughout by some factors, but this is a practical detail of no importance here. The right-hand side of (9.62) will need to be evaluated at every step, s times. At each stage i, the equation can be written as

$$\boldsymbol{M}\boldsymbol{k}_i = \boldsymbol{B}_i \tag{9.64}$$

where $\boldsymbol{B}_i$ is the evaluated right-hand side of (9.62). Thus, a linear system must be solved to obtain each $\boldsymbol{k}_i$. The matrix $\boldsymbol{M}$ will normally be either tri- or pentadiagonal or, in cases of simulations in more dimensions, will be rather sparse, so that either the Thomas algorithm or an offshoot of it, or a sparse solver, can be used, for efficiency. Also, the favoured Rosenbrock variants such as ROS2 or ROWDA3 have some zero coefficients, resulting in calculations that need not be repeated after the first stage, thus further increasing efficiency.

Having calculated all s $\boldsymbol{k}_i$ vectors, the RK formula is then used, here

$$\boldsymbol{C}_{n+1} = \boldsymbol{C}_n + \sum_{i=1}^{s} m_i \boldsymbol{k}_i \tag{9.65}$$

to compute the next concentration vector.

There are advantages, and also drawbacks, of this method. The advantages are great efficiency, stability and a smooth error response if ROS2 or ROWDA3 are used (see a study by Bieniasz [100], albeit not including ROS2), and the easy handling of time-dependent and/or nonlinear systems. No Newton iterations are required for nonlinear systems (but see below). The most serious drawback is that the method does not lend itself to problems with sharp initial transients, such as a potential step method; at least, not for the very first step, as pointed out by Bieniasz [100]. The reason is inconsistency. For example, in the Cottrell experiment, it is not possible to calculate a derivative $\boldsymbol{J}$ at the starting point, $T = 0$. One way to overcome this, taken by Bieniasz [100], is to start by invoking the boundary condition for $T > 0$ even initially. This can work, but can also lead to a persistent degradation of the results. In practical terms, for the Cottrell system, where all C_i, including C_0 should be unity at $T = 0$, we set $C_0 = 0$ at that time, and proceed. Note that this is exactly what is done in the explicit method. In order to avoid the degradation in accuracy (which is desirable for efficiency), the proper way is to use a different algorithm for the very first step, choosing one that expresses derivatives at $T = \delta T$, that is, an implicit method such as BI (Laasonen), perhaps coupled with extrapolation for improved accuracy. This might be considered defeating the Rosenbrock advantages, because if the system is nonlinear, one now needs some Newton iterations after all, and it could be argued that since one has programmed BI/extrapolation, one may as well proceed with it over the whole simulation. But Rosenbrock may be more efficient, so this is a compromise between programming effort and efficiency of computation.

Finally, it is to be noted that, when using Rosenbrock for an LSV simulation, one must be aware that the potential p at time T, at a given step goes to $p + \delta p$ at $T + \delta T$. It is the old value p that must be used in the boundary expressions, augmented by the α coefficients in higher stages. It is incorrect, in other words, to add δp to p at the beginning of the iteration loop.

9.4.1 An Example, the Birk-Perone System

There is an example program described in Appendix C, `BPROS`, applying Rosenbrock to the Birk-Perone system, in which we have both time-dependence and nonlinear equations. It is described in Chap. 2, pages 22–22. Equation (2.73), with boundary conditions, when semidiscretised using equal intervals in space, leads to the DAE system

$$
\begin{aligned}
0 &= C_0 \\
\frac{dC_1}{dT} &= \frac{1}{H^2}\left(C_0 - 2C_1 + C_2\right) - KC_1^2 \\
&\cdots \\
\frac{dC_i}{dT} &= \frac{1}{H^2}\left(C_{i-1} - 2C_i + C_{i+1}\right) - KC_i^2 \\
&\cdots \\
\frac{dC_N}{dT} &= \frac{1}{H^2}\left(C_{N-1} - 2C_N + C_{N+1}\right) - KC_N^2 \\
0 &= C_{N+1} - \frac{1}{1+KT}\,.
\end{aligned}
\tag{9.66}
$$

The last equation expresses the analytical solution for the time decay of the substrate. Here, then, we have selection matrix $\boldsymbol{S}$ as above (9.58), and the function $\boldsymbol{F}(T, \boldsymbol{C})$ is

$$
\boldsymbol{F}(T, \boldsymbol{C}) = \begin{bmatrix} C_0 \\ \frac{1}{H^2}\left(C_0 - 2C_1 + C_2\right) - KC_1^2 \\ \vdots \\ \frac{1}{H^2}\left(C_{N-1} - 2C_N + C_{N+1}\right) - KC_N^2 \\ C_{N+1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\frac{1}{1+KT} \end{bmatrix}, \tag{9.67}
$$

or regarding (9.59), $\boldsymbol{F}_C$, the derivative of $\boldsymbol{F}(T, \boldsymbol{C})$ with respect to $\boldsymbol{C}$,

$$
\boldsymbol{F}_C = \frac{1}{H^2}\begin{bmatrix} H^2 & & & & \\ 1 & -2(1+H^2KC_1) & 1 & & \\ & \ddots & & & \\ & 1 & -2(1+H^2KC_N) & 1 & \\ & & & H^2 \end{bmatrix} \tag{9.68}
$$

which contains concentration terms on the diagonal. For this reason, with this problem it is necessary to evaluate $\boldsymbol{F}_C$ at every step. Now noting the form of (9.62) and (9.63), and recalling $\lambda = \delta T/H^2$, it is convenient here to redefine $\boldsymbol{M}$ as

$$
\boldsymbol{M} = \frac{\boldsymbol{S}}{\gamma} - \delta T \boldsymbol{F}_C \tag{9.69}
$$

and after division by $-\lambda$, resulting in

$$
-\frac{\boldsymbol{M}}{\lambda} = \begin{bmatrix} H^2 & & & & \\ 1 & \left(-\frac{1}{\gamma\lambda} - 2(1+H^2KC_1)\right) & 1 & & \\ & \ddots & & & \\ & 1 & \left(-\frac{1}{\gamma\lambda} - 2(1+H^2KC_N)\right) & 1 & \\ & & & H^2 \end{bmatrix} \tag{9.70}
$$

(9.62) becomes

$$-\frac{\boldsymbol{M}}{\lambda}\boldsymbol{k}_i = -\frac{\delta T}{\lambda}\boldsymbol{F}\left(T+\alpha_i\delta T, \boldsymbol{C}+\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j\right) - \frac{\boldsymbol{S}}{\lambda}\sum_{j=1}^{i-1} c_{ij}k_j - \frac{\gamma_i\delta T^2}{\lambda}F_T(T,\boldsymbol{C}) \;, \tag{9.71}$$

where $\boldsymbol{F}_T$ remains to be defined. It is the time-derivative of $\boldsymbol{F}(T,\boldsymbol{C})$ and only contains one non-zero element:

$$\boldsymbol{F}_T = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\frac{K}{(1+KT)^2} \end{bmatrix} . \tag{9.72}$$

The above equation evaluates to a tridiagonal linear equation system, after some arrangement,

$$\begin{aligned} H^2 k_0 &= b_0 \\ k_0 + a_1 k_1 + k_2 &= b_1 \\ &\cdots \\ k_{l-1} + a_l k_l + k_{l+1} &= b_l \\ &\cdots \\ k_{N-1} + a_N k_N + k_{N+1} &= b_N \\ H^2 k_{N+1} &= b_{N+1} \end{aligned} \tag{9.73}$$

with

$$a_l = -\frac{1}{\gamma\lambda} - 2(1 + H^2 K C_l) \tag{9.74}$$

with the vector $\boldsymbol{b}$ arising in an obvious manner from the evaluation of the right-hand side of (9.71). This is the usual form for implicit systems, as seen in Chap. 8, page 121, albeit for unknown concentrations, here for unknown $\boldsymbol{k}$. Index l is used, i being reserved for the stage number here. The system is solved by the Thomas algorithm.

Execution of the program `BPROS` shows that it works well, attaining a relative accuracy of about 10^{-4} in about 100 steps of $\delta T = 0.01$, both with ROS2 and ROWDA3, the latter being slightly better (but using about 50% more CPU time).

9.5 FEM, BEM and FAM (briefly)

There is a class of methods called finite element method (FEM) and the related boundary element method (BEM), also called boundary integral element method (BIEM), and the finite analytical method (FAM). These will be

given very short shrift, in part because they constitute a large subject, many textbooks being devoted to FEM and BEM alone. The approach is usually to use ready program packages. The only member of the FEM group that will be described here is orthogonal collocation, which has its own section (see below).

Roughly, FEM consists of choosing regions in the simulation space, marked by node points, and fitting "trial" functions to the regions, in some optimal manner. What is considered optimal is defined in several different ways. With BEM, only points on the boundary are chosen and a function fitted to the space delimited by these points is optimised. Thus, BEM uses fewer points than FEM. Both methods appear to be highly efficient. FAM is similar to FEM, but instead of fitting an arbitrary function to the elements (in FEM, usually polynomials), local analytical solutions are sought for each of the elements.

Here are a few brief references to recent or key works in which these methods have been described as used in electrochemical simulations. The interested reader is urged to look these up and follow the references contained in them to the seminal works and text books. Of necessity, much work is left uncited here.

Ferrigno et al. [239] describe the use of FEM for steady state simulations of recessed, flush and protruding ultramicrodisk electrodes, giving a good description of FEM. The method was made adaptive by Nann (and Heinze) [407, 408], and Harriman et al. later published an extensive series of papers on adaptive FEM [287, 288, 289, 290, 291].

BEM might be thought of as best suited to steady state problems, and has been used for this, for example in corrosion simulations [64] and current distributions [198], but recently also for time-marching problems [457].

FAM has been investigated by Jin and Qian et al. [316, 317, 318, 454, 455, 456].

The newer method of Bortels et al., called multidimensional upwinding method (MDUM) should also be mentioned [127]. It was applied to a problem involving diffusion, convection and migration, both steady state and time-marching.

9.6 Orthogonal Collocation, OC

This is one of the variants of the finite element methods. The essence of orthogonal collocation (**OC**) is that a set of orthogonal polynomials is fitted to the unknown function, such that at every node point, there is an exact fit. The points are called collocation points, and the set of polynomials is chosen suitably, usually as Jacobi polynomials. The optimal choice of collocation points is to make them the roots of the polynomials. There are tables of such roots, and thus point placements, in Appendix A. The notable things here are the small number of points used (normally, about 10 or so will do), their

uneven spacing, crowding closer both at the electrode and (perhaps strangely) at the outer limit, and the fact that the outer limit is always unity. This is discussed below.

The method's historical origins are complex but electrochemists used OC first in 1970 [279], referring to an earlier work [576], for certain *ode*s. Caban and Chapman [159] then used OC to compute (steady state) current distributions, but the work most cited by later users of OC in electrochemistry is that of Whiting and Carr [571], who described its use in time-marching problems. They refer to the work of chemical engineers Villadsen and Stewart [563]. A later book [562] is a good source also, as is the chapter by Pons [446], drawing heavily on [562]. A number of electrochemists have published in the area, notably Pons and Speiser [444, 445], and later Speiser and coauthors; see the review by Speiser [523], for a complete list of references and a good description of OC, among other topics.

The possibly peculiar spacing of the collocation points, crowding close both at the electrode and at the outer diffusion limit, does not matter too much, and seems unnecessary. For example, using only five internal points (that is, five apart from zero and unity), they are placed at the values 0.047, 0.231, 0.5, 0.769, 0.953, a series that is symmetrical about the midway point at 0.500. This spacing has been circumvented by Yen and Chapman [580], using Chebyshev polynomials that open out towards the outer limit. Their work has apparently not been followed up.

OC is capable of high accuracy and efficiency. Some comparisons have been made with normal finite difference methods. Eddowes [217] found OC superior, while Magno et al. [376] found it inferior to plain EX with expanding intervals (this appears doubtful to the present author). Bieniasz and Britz [111] cast some doubt on OC, pointing out possible problems with the fit in between the collocation points, possibly leading to negative concentrations or (see below) errors in the current values computed from it. This was rebutted by Speiser [521], rather convincingly. The essence is that, if the concentration profile simulated is smooth (which it normally is), then the polynomials will be well behaved in between points and no such problems will be encountered. As is seen below, implicit boundary values can easily be accommodated, and by the use of spline collocation [303, 443, 453], homogeneous chemical reactions of very high rates can be simulated. This refers to the static placement of the points. Having, for example, the above sequence of points for five internal points, the point closest to the electrode is at 0.047. This will be seen, below, to be in fact further from the electrode than it seems, because of the way that distance X is normalised so that, for very fast reactions that lead to a thin reaction layer, there might not be any points within that layer. Spline collocation thus takes the reaction layer and places another polynomial within it, while the region further out has its own polynomial. The two polynomials are designed such that they join smoothly,

both with the same gradient at the join. This will not be described further here.

For the description of how OC works, assume for simplicity a single substance. The diffusion equation, including a homogeneous reaction, is

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} + f(c) \tag{9.75}$$

with $f(c)$ being the homogeneous reaction term, left unspecified. The equation is written in dimensioned form for a reason. In OC, the space axis is normalised in a manner different from the usual. Instead of normalising by the diffusion layer thickness δ, as outlined in Chap. 2, Sect. 2.3, it is here normalised by the total diffusion space width L, so that the range is $0 \leq X \leq 1$, in order to fit in with the range of the polynomial. The value of L depends on the experiment being simulated, and will be a multiple of $\sqrt{D\tau}$, τ being some characteristic time. As explained in previous chapters, τ might be the duration of the experiment for pulse experiments or the length of time taken by the potential, for a linear sweep, to change by one dimensionless potential unit. In general, it is given by

$$L = f\sqrt{D\tau} \tag{9.76}$$

and thus,

$$X = x/(f\sqrt{D\tau})\ . \tag{9.77}$$

Time and concentration are normalised as usual (Sect. 2.3), and this leads to

$$\frac{\partial C}{\partial T} = \frac{1}{f^2}\frac{\partial^2 C}{\partial X^2} + F(C)\ , \tag{9.78}$$

where we now have the factor $1/f^2$, (and $F(C)$ is the dimensionless form of the rate equation for the homogeneous reaction term). The new factor is usually written as β, and is often discussed as an arbitrarily adjustable parameter. But it is not; it must be determined by L, which is known for a given experiment. For, say, the Cottrell experiment, $f = 6$, while for a linear or cyclic sweep, it is $6\sqrt{T}$, with T being the total number of potential (or time) units swept during the experiment. If one takes β to be arbitrary, one might either, by making it too small (*i.e.* L too large), simulate a far too wide diffusion space and thus degrade the resolution near the electrode or, by making it too large, simulate in a confined space, so that the outer boundary concentration cannot be taken as constant throughout. The choice of β is always rational.

As an aside, there have been some interesting attempts to make the diffusion space variable with time and to normalise by that variable. Yen and Chapman [580] used this, and Urban and Speiser [550]. The diffusion equation then normalises to a rather more complicated form, sometimes into a plain second-order *ode*, or in other cases, into a form including time-dependent

terms in $\partial C/\partial X$. Results [550] appeared to be very good. This has apparently not been followed up, but perhaps it should be.

Now for the description of how OC works. Assume a number $N+2$ points situated at $X_0, X_1, \ldots, X_N, X_{N+1}$, and X lying in the interval $[0,1]$ by the normalisation described above. The points are chosen, following the work of Whiting and Carr [571], as the roots of shifted Jacobian polynomials with parameters as given in the Tables A.3–A.5 in Appendix A. The tabled values were computed using the subroutine `JCOBI` mentioned in Appendix C. The concentration profile is approximated by the polynomial $P(X)$,

$$C(X) \approx P(X) = \sum_{j=0}^{N+1} b_j X^j \tag{9.79}$$

where b_j are coefficients which, as it happens, we never need to find. The OC method assumes that $P(X)$ exactly fits $C(X)$ at each of the collocation points. We have some derivatives,

$$\frac{dC}{dX} = \frac{d}{dX} P(X) = \sum_{j=0}^{N+1} j b_j X^{j-1} \tag{9.80}$$

and

$$\frac{d^2C}{dX^2} = \frac{d^2}{dX^2} P(X) = \sum_{j=0}^{N+1} j(j-1) b_j X^{j-2} \,. \tag{9.81}$$

Equations (9.79), (9.80) and (9.81) can be written out for every value of X_j, leading to the systems of equations

$$\begin{aligned} C_0 &= \sum_{j=0}^{N+1} b_j \; X_0^j \\ C_1 &= \sum_{j=0}^{N+1} b_j \; X_1^j \\ &\ldots \\ C_{N+1} &= \sum_{j=0}^{N+1} b_j \; X_{N+1}^j \end{aligned} \tag{9.82}$$

for the concentrations themselves,

$$\begin{aligned}
\frac{dC_0}{dX} &= \sum_{j=0}^{N+1} j\ b_j\ X_0^{j-1} \\
\frac{dC_1}{dX} &= \sum_{j=0}^{N+1} j\ b_j\ X_1^{j-1} \\
&\dots \\
\frac{dC_{N+1}}{dX} &= \sum_{j=0}^{N+1} j\ b_j\ X_{N+1}^{j-1}
\end{aligned} \tag{9.83}$$

for the first derivatives with respect to X, and

$$\begin{aligned}
\frac{d^2C_0}{dX^2} &= \sum_{j=0}^{N+1} j(j-1)\ b_j\ X_0^{j-2} \\
\frac{d^2C_1}{dX^2} &= \sum_{j=0}^{N+1} j(j-1)\ b_j\ X_1^{j-2} \\
&\dots \\
\frac{d^2C_{N+1}}{dX} &= \sum_{j=0}^{N+1} j(j-1)\ b_j\ X_{N+1}^{j-2}
\end{aligned} \tag{9.84}$$

for the second derivative. These equations are now written in matrix form:

$$\begin{aligned}
\boldsymbol{C} &= \boldsymbol{Q}\boldsymbol{b} \\
\frac{d\boldsymbol{C}}{dX} &= \boldsymbol{R}\boldsymbol{b} \\
\frac{d^2\boldsymbol{C}}{dX^2} &= \boldsymbol{S}\boldsymbol{b}\ ,
\end{aligned} \tag{9.85}$$

with $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ obvious from the systems above.

So far, we have the set of coefficients, vector $\boldsymbol{b}$, which we do not know. These are now eliminated by expressing the first equation of the set (9.85) explicitly for $\boldsymbol{b}$:

$$\boldsymbol{b} = \boldsymbol{Q}^{-1}\boldsymbol{C} \tag{9.86}$$

and substituting for it in the other two, giving

$$\frac{d\boldsymbol{C}}{dX} = \boldsymbol{R}\boldsymbol{Q}^{-1}\boldsymbol{C} = \boldsymbol{V}\boldsymbol{C} \tag{9.87}$$

and

$$\frac{d^2\boldsymbol{C}}{dX^2} = \boldsymbol{S}\boldsymbol{Q}^{-1}\boldsymbol{C} = \boldsymbol{W}\boldsymbol{C} \tag{9.88}$$

with

$$\boldsymbol{V} = \boldsymbol{R}\boldsymbol{Q}^{-1} \tag{9.89}$$

and

$$\boldsymbol{W} = \boldsymbol{S}\boldsymbol{Q}^{-1} \, . \tag{9.90}$$

The above presupposes that $\boldsymbol{Q}$ is invertible, and this is the case, as the system (9.82) has no linearly dependent pairs of lines. The interesting thing is that $\boldsymbol{V}$ and $\boldsymbol{W}$ can be precomputed, for a given N, once and for all, simply from the Jacobi roots. Equation (9.88) can now be inserted in (9.78), to produce

$$\frac{\partial \boldsymbol{C}}{\partial T} = \frac{1}{f^2}\boldsymbol{W}\boldsymbol{C} + F(\boldsymbol{C}) \tag{9.91}$$

or, the set of $N + 2$ equations

$$\begin{aligned}
\frac{\partial C_0}{\partial T} &= \frac{1}{f^2}(W_{0,0}C_0 + W_{0,1}C_1 + \cdots + W_{0,N}C_N + W_{0,N+1}C_{N+1}) + F(C_0) \\
\frac{\partial C_1}{\partial T} &= \frac{1}{f^2}(W_{1,0}C_0 + W_{1,1}C_1 + \cdots + W_{1,N}C_N + W_{1,N+1}C_{N+1}) + F(C_1) \\
\frac{\partial C_2}{\partial T} &= \frac{1}{f^2}(W_{2,0}C_0 + W_{2,1}C_1 + \cdots + W_{2,N}C_N + W_{2,N+1}C_{N+1}) + F(C_2) \\
&\cdots \\
\frac{\partial C_N}{\partial T} &= \frac{1}{f^2}(W_{N,0}C_0 + W_{N,1}C_1 + \cdots + W_{N,N}C_N + W_{N,N+1}C_{N+1}) \\
&\quad + F(C_N) \\
\frac{\partial C_{N+1}}{\partial T} &= \frac{1}{f^2}(W_{N+1,0}C_0 + W_{N+1,1}C_1 + \cdots + W_{N+1,N}C_N \\
&\quad + W_{N+1,N+1}C_{N+1}) + F(C_{N+1}) \quad .
\end{aligned} \tag{9.92}$$

This is written out in order to make the next point. The first and last equation in the set are superfluous, because the boundary concentrations C_0 and C_{N+1} are not subject to diffusion changes, but to other conditions. Also, where the boundary values appear in the other equations, they must be replaced with what we can substitute for them. The outer boundary value, C_{N+1}, is (almost always) equal to the initial bulk concentration C^*, usually equal to unity in its dimensionless form. This means that the last term in each equation separates out as a constant term and makes for a constant vector $\left[W_{1,N+1}C^* \;\; W_{2,N+1}C^* \;\; \ldots \;\; W_{N,N+1}C^*\right]^T$, which will be called $\boldsymbol{Z}$ here. The concentration at the electrode C_0 is handled according to the boundary condition. For Cottrell, for example, it is set to zero throughout and thus simply drops out of the set. For other conditions, for example constant current or an irreversible reaction, a gradient G is involved, as described in Chap. 6. In that chapter, the gradient was expressed as a possibly multipoint approximation,

but here we have a better device: the use of matrix $\boldsymbol{V}$, applying it to obtain $G = dC/dX(X = 0)$:

$$G = \sum_{j=0}^{N+1} V_{0,j} C_j \ . \tag{9.93}$$

This is written explicitly for C_0,

$$C_0 = \beta_1 C_1 + \beta_2 C_2 + \cdots + \beta_N C_N + (\beta_{N+1} C^* - G/V_{0,0}) \tag{9.94}$$

where $\beta_j = -V_{0,j}/V_{0,0}$. This can be substituted into the N equations of the set, which adds terms to the W-coefficients and the constant term (the one in brackets on the right-hand side of (9.94)) to the constant vector $\boldsymbol{Z}$. We thus obtain a smaller $N \times N$ equation set,

$$\frac{\partial \boldsymbol{C}}{\partial T} = \frac{1}{f^2} \boldsymbol{W'C} + \boldsymbol{Z} + F(\boldsymbol{C}) \ . \tag{9.95}$$

For more than one species, the development is clear, based on Chap. 6, leading to larger systems of equations.

The big advantage here is that the matrix $\boldsymbol{W}$ is always the same for a given N. If one works always with some favourite value, such as 10 (a good value), then one needs to compute $\boldsymbol{W}$, and indeed $\boldsymbol{V}$, only once, and use it as input data thereafter.

We have now arrived at the point where a choice needs to be made of how to proceed with the simulation. Note that (9.95) is in fact of the same form as that obtained when using MOL, being of the same form as the set (9.17), but with more coefficients on every line. From here on, one can use a variety of methods to do the time-march. All of the methods considered in earlier chapters, and this chapter, can be used. As with FEM, however, there is a certain tradition here, for using ready-made *ode* solvers. Villadsen and Michelsen [562] use an implicit Runge-Kutta algorithm devised by Caillaud and Padmanabhan [160], implementing it in their routine `STIFF3`. The routine was reproduced by Pons [446] in his chapter, mentioning that he too finds the method of Caillaud and Padmanabhan best. Whiting and Carr [571] wrote their own solver, based on a predictor-corrector algorithm. Speiser [523] describes several other subroutine packages, such as `DDEBDF` arising from the work of Gear [263] or `LSODE` of Hindmarsh [304]. Bieniasz and Britz [111] found the routine `STINT` [467] more efficient than `STIFF3`.

Another possible approach, apparently not taken by any electrochemical simulator, is to render the equation set into a DAE set. Instead of substituting for C_0 as described above, one replaces the first equation of (9.92) by the algebraic equation for the boundary condition, and uses one of the available DAE packages to solve the system.

Both the approaches described above, that is, substituting for boundary concentrations, or adding algebraic equations to express boundary conditions, can be applied to more complex mechanisms involving more than one species,

including coupled systems. With the latter, there is probably not much to be gained by the Rudolph method, because of the number of coefficients on each line.

The present author has used the simple second-order extrapolation technique to proceed from (9.95), and this simple approach led to highly accurate results. Using just 10 points (8 internal points) and 100 steps in time for a Cottrell simulation, the current was in error by only 0.01%. Using only 3 internal points, there was a 10% error in the current. These are remarkable results. For this, second-order extrapolation was used in combination with BI.

9.6.1 Current Calculation with OC

The normalised current, that is the gradient G, is given by using matrix $\boldsymbol{V}$ and (9.87). The operation returns gradients at all collocation points, and one just takes the first of these, which refers to $X = 0$. Alternatively, one can multiply just the top row of $\boldsymbol{V}$ with the concentration vector $\boldsymbol{C}$, which gives $dC/dX(X = 0)$ directly. Note that this is not our usual G yet, because of the way X is normalised here. Regarding (9.77), clearly,

$$G = \frac{1}{f}\frac{dC}{dX}(X = 0) \; . \tag{9.96}$$

Alternatively, one can simply work with the X-scaling as it is, and change the analytical solution correspondingly. For example, for the Cottrell experiment, with the usual normalisation, the gradient G at time T has the analytical solution $1/\sqrt{\pi T}$ (see (2.44) on page 18) while, with the normalisation as used in OC (9.77), the analytical solution for the gradient at the electrode ((2.36, page 16) becomes

$$G = \left.\frac{\partial C}{\partial X}\right|_{X=0} = f\frac{1}{\sqrt{\pi T}} \; . \tag{9.97}$$

In the case of LSV, however, not dividing by the factor f would lead to currents that are hard to compare with tabled values or values one expects.

9.6.2 A Numerical Example

For those wanting to try OC, here is a guide for checking the work. This follows the example given by Whiting and Carr [571].

Assume a Cottrell simulation and the use of only five points, giving just three internal points. From Table A.3, this places the internal points at the positions (0.1127, 0.5000, 0.8873), here presenting fewer digits than in the table. Using equation sets (9.82)–(9.84) and the definitions (9.85), we then have

$$\boldsymbol{Q} = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.0000 & 0.1127 & 0.0127 & 0.0014 & 0.0002 \\ 1.0000 & 0.5000 & 0.2500 & 0.1250 & 0.0625 \\ 1.0000 & 0.8873 & 0.7873 & 0.6986 & 0.6198 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \end{bmatrix} \tag{9.98}$$

$$\boldsymbol{R} = \begin{bmatrix} 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.2254 & 0.0381 & 0.0057 \\ 0.0000 & 1.0000 & 1.0000 & 0.7500 & 0.5000 \\ 0.0000 & 1.0000 & 1.7746 & 2.3619 & 2.7943 \\ 0.0000 & 1.0000 & 2.0000 & 3.0000 & 4.0000 \end{bmatrix} \tag{9.99}$$

and

$$\boldsymbol{S} = \begin{bmatrix} 0.0000 & 0.0000 & 2.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 2.0000 & 0.6762 & 0.1524 \\ 0.0000 & 0.0000 & 2.0000 & 3.0000 & 3.0000 \\ 0.0000 & 0.0000 & 2.0000 & 5.3238 & 9.4476 \\ 0.0000 & 0.0000 & 2.0000 & 6.0000 & 12.0000 \end{bmatrix} . \tag{9.100}$$

From these, using (9.89) and (9.90), we obtain

$$\boldsymbol{V} = \begin{bmatrix} -13.0000 & 14.7883 & -2.6667 & 1.8784 & -1.0000 \\ -5.3238 & 3.8730 & 2.0656 & -1.2910 & 0.6762 \\ 1.5000 & -3.2275 & 0.0000 & 3.2275 & -1.5000 \\ -0.6762 & 1.2910 & -2.0656 & -3.8730 & 5.3238 \\ 1.0000 & -1.8784 & 2.6667 & -14.7883 & 13.0000 \end{bmatrix} \tag{9.101}$$

and

$$\boldsymbol{W} = \begin{bmatrix} 84.0000 & -122.0632 & 58.6667 & -44.6035 & 24.0000 \\ 53.2379 & -73.3333 & 26.6667 & -13.3333 & 6.7621 \\ -6.0000 & 16.6667 & -21.3333 & 16.6667 & -6.0000 \\ 6.7621 & -13.3333 & 26.6667 & -73.3333 & 53.2379 \\ 24.0000 & -44.6035 & 58.6667 & -122.0632 & 84.0000 \end{bmatrix} . \tag{9.102}$$

Matrix $\boldsymbol{V}$ is needed to generate G, and $\boldsymbol{W}$ is now stripped of its outer frame to produce (9.95),

$$\frac{\partial \boldsymbol{C}}{\partial T} = \frac{1}{f^2} \begin{bmatrix} -73.3333 & 26.6667 & -13.3333 \\ 16.6667 & -21.3333 & 16.6667 \\ -13.3333 & 26.6667 & -73.3333 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} + \begin{bmatrix} 6.7621 \\ -6.0000 \\ 53.2379 \end{bmatrix} + F(\boldsymbol{C}) . \tag{9.103}$$

The above set of *ode*s is now solved, choosing some algorithm. Nothing has been specified about the homogeneous chemical reaction function $F(\boldsymbol{C})$, but it will add terms to the matrix $\boldsymbol{W'}$ when specified. After the time derivative is discretised in some way, the equation can be rearranged into the same form as described in Chap. 8 and solved using the same methods or, as mentioned above, solved using a professional *ode* or DAE solver.

9.7 Eigenvalue-Eigenvector Method

Yet another, quite different, approach to solving a system of *ode*s, such as one obtains as an intermediate step when using, for example, MOL or OC, is the eigenvalue-eigenvector method. Its use for electrochemical simulations was described in two papers in 1989 and 1990 [255, 332]. The method has some drawbacks, and does not appear to have seen much use since these two papers. It does have one unique feature: there is no discretisation of time. A solution is generated by the algorithm, at any chosen time. So, although the method may at times be fairly inefficient, if one wants a current or concentrations at only one or a few time points, this could be faster than a time march with the usually small time intervals.

The method is also described rather clearly by Smith [514], whose description will be followed here.

The method starts with a system of *ode*s, represented as in (9.54),

$$\frac{d\boldsymbol{C}}{dT} = \boldsymbol{A}\boldsymbol{C} \,, \tag{9.104}$$

simplified here so that the vector coming from boundary conditions is not included. It can be included but then the argument is less focussed. There is only one boundary condition,

$$\boldsymbol{C}(T=0) = \boldsymbol{C}(0) \,. \tag{9.105}$$

Instead of now discretising the left-hand side of the equation in some way (explicit BI, CN, etc) and stepping forward in time by small time intervals, the equation is solved analytically; the solution is

$$\boldsymbol{C}(T) = \exp(T\boldsymbol{A})\,\boldsymbol{C}(0) \,. \tag{9.106}$$

This has an exponential of a matrix. It is defined in terms of the expansion of the exponential function, see for example Smith [514, p. 134–5]. Now, the usual eigenvalue-eigenvector equation can be written in compact form,

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{D} \,, \tag{9.107}$$

where $\boldsymbol{D}$ is the diagonal matrix containing all the eigenvalues and $\boldsymbol{X}$ is the matrix containing the eigenvectors corresponding to the eigenvalues. $\boldsymbol{X}$ is also called the modal matrix of $\boldsymbol{A}$. $\boldsymbol{X}$ and $\boldsymbol{D}$ can be computed by available subroutines, and Friedrichs et al. describe very efficient ways of calculating them. The method rests crucially on these values.

Equation (9.107) can be written explicitly for $\boldsymbol{D}$:

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{D} \,. \tag{9.108}$$

Smith also shows that it follows from this that

$$\boldsymbol{X}^{-1}\exp(\boldsymbol{A})\boldsymbol{X} = \exp(\boldsymbol{D}) \tag{9.109}$$

which will be useful below.

Let a new matrix $\boldsymbol{Y}(T)$ be defined as

$$\boldsymbol{C}(T) = \boldsymbol{X}\boldsymbol{Y}(T) \tag{9.110}$$

with the time-dependent vector $\boldsymbol{Y}$ as yet unknown. We can however solve for $\boldsymbol{Y}(0)$ by setting $T = 0$ in (9.110), since we know $\boldsymbol{X}$ and the initial condition $\boldsymbol{C}(0)$. Once we have, for any other T, the vector $\boldsymbol{Y}(T)$, it can be used, via (9.110), to compute the desired vector $\boldsymbol{C}$, by multiplication with $\boldsymbol{X}$.

Combining (9.106) with (9.110), we can write

$$\boldsymbol{X}\boldsymbol{Y}(T) = \exp(T\boldsymbol{A})\,\boldsymbol{X}\boldsymbol{Y}(0) \tag{9.111}$$

and, multiplying by $\boldsymbol{X}^{-1}$,

$$\boldsymbol{Y}(T) = \boldsymbol{X}^{-1}\exp(T\boldsymbol{A})\boldsymbol{X}\boldsymbol{Y}(0)\,. \tag{9.112}$$

Equation (9.109) allows us to write this as

$$\boldsymbol{Y}(T) = \exp(T\boldsymbol{D})\boldsymbol{Y}(0)\,. \tag{9.113}$$

Here we have an exponential of the matrix $\boldsymbol{D}$. The matrix is zero except on the diagonal (containing the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$), and, as Smith proves, this and the definition of a matrix exponential lead to the simple result that

$$\exp(T\boldsymbol{D}) = \begin{bmatrix} \exp(T\lambda_1) & & & & \\ & \exp(T\lambda_2) & & & \\ & & \exp(T\lambda_3) & & \\ & & & \ddots & \\ & & & & \exp(T\lambda_N) \end{bmatrix}, \tag{9.114}$$

so that finally the solution is

$$\begin{bmatrix} C_1(T) \\ C_2(T) \\ \vdots \\ C_N(T) \end{bmatrix} = \boldsymbol{X} \begin{bmatrix} Y_1(0)\exp(T\lambda_1) \\ Y_2(0)\exp(T\lambda_2) \\ \vdots \\ Y_N(0)\exp(T\lambda_N) \end{bmatrix}. \tag{9.115}$$

This entails matrix multiplication, and Smith suggests an approximation. The eigenvalues λ_j are negative and of increasing magnitude as j increases, so if T is not too small, the very first term of the right-hand vector is dominating, and can in itself produce a good approximation to the solution. This has the drawback that no reliable solution can be found for small T, and Friedrichs

et al. [255] do not suggest this approximation but rather, a more efficient way to calculate the eigenvalues and -vectors. Their solutions are fairly accurate.

As mentioned, the procedure has the advantage that the time variable T is part of the solution expression, so that if solutions at only a few time values, or even just one such T, are sought, the method might be competitive with the more usual time-marching schemes. Also, although the above description has been simplified by leaving out the boundary condition vector in (9.104), its addition still leaves the method intact. As shown in the second paper [332], LSV simulations and quasireversible systems can be handled. For some reason, however, the method has not seen any use in electrochemistry since these two seminal papers.

9.8 Integral Equation Method

In the specific case of LSV or CV (which however these days comprises a large proportion of electrochemistry experiments carried out), there is an alternative method for computing the current, and surface concentrations. Often, this is all one needs. Instead of starting with the diffusion equation(s), one begins with an attempt at an analytical solution, by means of the Laplace transformation. See the standard texts such as Bard and Faulkner [74] or Galus [257] for a description of the procedure. With LSV/CV, one then seems to arrive at a dead end, in the form of an integral equation. Regarding the description of the mathematics of LSV in Chap. 2, from page 25, and taking the example of a simple reversible system, the result is the equation, for the normalised current (Randles-Ševčík function $\chi(z)$),

$$\int_0^{at} \frac{\chi(z)}{\sqrt{at-z}} dz = \frac{1}{1+\xi\Theta S(at)} \ , \tag{9.116}$$

where $\xi = (D_O/D_R)^{\frac{1}{2}}$, $\Theta = \exp\left\{\frac{n\mathcal{F}}{\mathcal{R}T}(E_i - E^0)\right\}$ and $S(at) = \exp(-at)$, at being the normalised time variable and E_i the initial potential at the start of the sweep. The equation is that arising from the simple reversible system, but other systems lead to equations of similar form, all Volterra equations.

In descriptions of this problem, the names of Randles [460] and Ševčík [505] are prominent. They both worked on the problem and reported their work in 1948. Randles was in fact the first to do electrochemical simulation, as he solved this system by explicit finite differences (and using a three-point current approximation), referring to Emmons [218]. Ševčík attempted to solve the system analytically, using two different methods. The second of these was by Laplace transformation, which today is the standard method. He arrived at (9.116) and then applied a series approximation for the current. Galus writes [257] that there was an error in a constant. Other analytical solutions were described (see Galus and Bard and Faulkner for references), all in the form of series, which themselves require quite some computation to evaluate.

So the direct approximation of equations like (9.116) was an obvious step. This was taken in the classic paper of Nicholson and Shain [417], and continued, by Nicholson [415] and Nicholson and Olmstead [416] for systems other than the simple reversible. These used what is called the Huber method [311], integrating by summing a number of intervals into which the limit *at* is divided. If there are N intervals, the Huber method gives rise to a triangular system of N simultaneous equations, which requires of the order N^2 operations for the solution. Bieniasz devised a better method [89,91] that requires only of the order of N operations. In an example using both the Huber and his new method, the Huber method required 39 min for a computation, while the improved method did the same in 0.13 min. This method is built into Bieniasz' simulation package `ELSIM` [92], among others. In his 1992 paper, he also points out the mathematical relation between the integral equation above, and the process of semi-integration, described by Oldham [425], for which there are also more and less efficient algorithms. Bieniasz lists [91] a table of the forms of the integral equation for a number of systems. More recently, Mirčeski has published an approximation to the integral, separating the current function out as a sum, which he claims is an efficient method of solving these equations.

We do not go into any detail of the integration methods here, as it seems that direct finite difference methods are preferable.

9.9 The Network Method

Since about 1989, Horno and coworkers have published a series of papers on their "network thermodynamic method" of simulation. Only a few of these will be cited here. In the first, the 1989 work, the method is described [309], and again in 1992–4 [271,305,306], adding cyclic voltammetry. In the 1994 paper [305], there is a good description of the method, and an indication how it can be adapted to a multitude of different electrochemical systems. A Chinese group has also used this method [205,208,209,210].

It is all done by modelling derivatives, fluxes and homogeneous chemical reactions as electrical elements and current sources, and applying Kirchhoff's Law to them. After conversion to an analogue of an electric circuit, the standard package SPICE or PSPICE then does the rest [406,549], using Gear's package for solving *ode*s [263]. The main work for the simulator is thus the translation of the governing electrochemical equations into an electrical network and specifying it to the packages.

Very briefly, basing the description on [305], the diffusion-reaction equation, of the form of (9.75) is semidiscretised as in MOL, to

$$\frac{dC}{dt} = \frac{D}{h^2}\left(C_{i-1} - C_i\right) + \frac{D}{h^2}\left(C_i - C_{i+1}\right) + f(C)\,. \tag{9.117}$$

Then both sides are multiplied by the spatial interval h, and the result expressed, term by term, as

$$J_{\gamma i} = J_{i-1} - J_i + J_{Gi} \tag{9.118}$$

with

$$J_{\gamma i} = h\frac{dC_i}{dt} \,, \tag{9.119}$$

regarded as a capactive flux due to "capacity" h and "voltage" C_i. The two terms J_i and J_{i-1} are regarded as resistive fluxes due to "resistance" h/D and again "voltage" C, and finally J_{Gi} corresponds to the homogeneous reaction term, seen as a current source, which might depend on one or more "voltages" (concentrations), depending on the reaction. These elements are then arrayed in a suitable manner in a ladder network, and the input to SPICE or PSPICE is designed for that. It seems that this process of translation into a sequence of specifications to SPICE (which Horno indeed calls a program) is the main work, and appears to the present author to be rather indirect and cumbersome. Probably workers familiar with the method, as the Horno school is, have a different view. The method has been successfully applied to such difficult systems as catalytic second-order reactions [307], migration [405], oscillating reaction-diffusion systems [308], the square scheme [258] and lately, steady state colloidal systems, solving for potential fields [369]. The list of papers is only partial.

The method has not taken on. One paper [168] reports the use of PSPICE, but for simulating actual resistance in an electrolyte, modelled as a resistance network. This is quite a different application, and much more directly relevant.

9.10 Treanor Method

In a paper reporting the results of some simulations of diffusion of hydrogen into palladium [582], the authors describe their method of solution as the Treanor method. This is described in a few texts [314, 351] and goes back to a paper by Treanor in 1966 [548].

The method is one way to handle a stiff set of *ode*s, and is an extension of fourth-order explicit Runge-Kutta. The function to be solved is approximated over the next time interval by a combination of a linear function of the dependent variable and a quadratic function of time (assuming that it is strongly time-dependent) and this increases the accuracy and stability of the fourth-order Runge-Kutta method considerably. Today, however, we have other methods of dealing with stiff sets of *ode*s, so this method might be said to have outlived its usefulness.

9.11 Monte Carlo Method

Diffusion is at base a process due to randomly moving particles, so it might be logical to model or simulate it as such. This has been done in a few works. Fanelli et al. [226, 227] thus simulated adsorption processes, using a method described earlier by Voss and Tomkiewicz [566]. Licht et al. [363] simulated concentration profiles around arrays of generator-collector microbands. Borkowski and Stojek simulated a CV at a microelectrode [126]. Up to 35000 particles were let loose to do a random walk. The result was a very rough but recognisable CV. Baur and Motsegood simulated a pair of coplanar disks [84]. This is interesting but hardly recommended, having no obvious advantages over other methods and being presumably somewhat time-consuming, although Baur and Motsegood argue for its use.

10 Adsorption

In this chapter, it is shown how to simulate the adsorption of a substance, not taking into account any electrochemical reactions the substance may undergo. That is, only the adsorption itself is dealt with here. In Chap. 2, Sect. 2.5, some theory is presented, laying the groundwork for the simulation. It is noted there that adsorption may be controlled by transport and the adsorption isotherm, in which case there is equilibrium at all times between the solution and surface phases; or that the adsorption step itself may limit the rate of adsorption. In this latter case, there are rate constants whose values must be known. In both cases, for isotherms more complicated than the Henry isotherm (2.104), nonlinear terms will enter the equations to be solved in a simulation.

Simulation of adsorption kinetics is not given as much attention as electron transfer, but some work has been done over the years. Analytical solutions are few and far between, as mentioned in Chap. 2, practically existing only for the Henry isotherm. So, as for electron transfer, simulation is needed. Rampazzo [459] was one of the first, using a numerical solution of the Volterra integral equation describing the adsorption kinetics. Flanagan et al. [248] mention nonlinear terms in a simulation, as do Miller and coworkers [395,396,397,398], Lovrić et al. [371]. Britz et al. [145] considered nonlinear isotherms as part of the boundary conditions in an implicit simulation and, more recently, Hsu et al. [310] modelled adsorption at an air-water interface. Bieniasz introduced the concept of an "interfacial species" in his work [98], and has incorporated adsorption kinetics in his program package `ELSIM` [92]. These are just a few examples taken out of a wider literature.

Rather than the integral equation approach of Rampazzo [459], the direct simulation from the transport equations is used here. In order to obtain a certain surface concentration Γ or fractional coverage θ, the substance in question must first arrive at the electrode, by some transport process. As was shown in Chap. 2, the normalised equation describing the accumulation of substance at the electrode is

$$\frac{d\theta}{dT} = KG \tag{10.1}$$

with K being the normalising collection $c^*\sqrt{D\tau}/\Gamma_m$. This equation must be supplemented by another, describing the relation between the coverage θ

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 189–192 (2005)
www.springerlink.com

and the concentration C_0 in solution at $X = 0$. This is either an equation involving an adsorption isotherm or one involving adsorption rates. For both these cases, explicit or implicit methods can be used.

There is an extreme case – that of very strong adsorption (b is large), leading to the approximate condition $c_0 \approx 0$ (all t). This is just like the electrochemical purely diffusion limited potential step case, for which we have the solution $G(T)$, (2.44) and (2.26). G can now be inserted into (10.1), and simple integration then gives:

$$\theta(T) = \frac{2c_b\sqrt{D\tau}}{\Gamma_{max}\sqrt{\pi}}\sqrt{T}\ . \tag{10.2}$$

This was also solved for the dropping mercury electrode by Koryta in 1953 [342]. Other cases, either fast adsorption with consideration of isotherms, or rate-limiting adsorption, will now be described. In all cases below, a new variable θ must be added to the unknowns, vector $\boldsymbol{C}$. Conveniently, we make it the first element of all the unknowns.

10.1 Transport and Isotherm Limited Adsorption

For this case, we have, apart from the usual diffusion equations, two boundary condition equations relating C_0 and θ. They are

$$\begin{aligned} BC_0 &= I(\theta) \\ \frac{d\theta}{dT} &= KG \end{aligned} \tag{10.3}$$

where, as mentioned above, K comes from the collection of parameters that go into the normalisation, and G, as usual, is the concentration gradient at $X = 0$. To this small set must now be added the N discretised equations describing the diffusion of the substance in solution.

There are now several choices of method. The simplest may be the explicit method. Using this, one starts at time T, where we know all values, and use them to proceed to the new time $T + \delta T$. First, one recalculates all $C_i, i = 1 \ldots N$. Parallel with this, from the value of G, one calculates a new θ. Discretising the second equation of the set (10.3) and expanding G as usual as an n-point approximation leads to

$$\theta' = \theta + \delta T K \sum_{i=0}^{n-1} \beta_i C_i\ . \tag{10.4}$$

Then, the new value of θ is used in the isotherm equation to recalculate C_0. This is simple, but has the drawback of poor accuracy and the limit on the λ factor. Clearly, an implicit method is preferable, such as BI with extrapolation, for example. The diffusion part of the whole set of equations

will depend on the placing of the points in space, as described in Chap. 8, for example using the general three-point (8.8) on page 120, or a multi-point form such as (8.31), page 124. These lead to the usual system as (8.11) or its multipoint relative (8.33) on pages 121 and 124. The first step is to do the backward Thomas scan as described in that chapter, and to apply the u-v procedure, resulting in a set of linear expressions for the first n (as yet unknown) concentration values, in terms of C'_0,

$$C'_i = u_i + v_i C'_0 \; . \tag{10.5}$$

This can now be substituted into the G approximation above (10.4) and now discretising the whole (10.4) according to the BI method (only new values used), we have

$$\theta' = \theta + K\delta T \left(\sum_{i=0}^{n-1} \beta_i (u_i + v_i C'_0) \right) \tag{10.6}$$

which can be rearranged in terms of the two remaining unknowns θ' and C'_0, into

$$\theta' = P + QC'_0 \tag{10.7}$$

with P and Q obvious from (10.6). This must now be combined with the first boundary condition of (10.3). If the Henry isotherm holds, this is simply

$$BC'_0 = \theta' \tag{10.8}$$

and the solution follows easily. If, on the other hand, a nonlinear isotherm such as Langmuir or Frumkin isotherm holds, we have a nonlinear pair of equations. This can be solved using the Newton method. It will normally be aided by the fact that at a given step, both θ and C_0 change only a little, so the Newton process will probably converge rapidly. Details are left to the reader.

An obvious alternative choice of method, given the probably nonlinear form of the isotherm boundary condition is to use a Rosenbrock method. Then, the two boundary conditions are simply the first two equations in a whole DAE set, the first of the pair (10.3) being an algebraic equation, the second an *ode*. The Rosenbrock method is described in Chap. 9, Sect. 9.4 starting on page 167.

10.2 Adsorption Rate Limited Adsorption

If adsorption itself is a slow process, then rate equations for that process apply, as outlined in Chap. 2, from page 31. As with the isotherm-dependent boundary conditions, we may have nonlinear equations, such as (2.121). The boundary conditions, inserting (2.121) into (2.117), are

$$\frac{d\theta}{dT} = V_f - V_b$$
$$\frac{d\theta}{dT} = KG \,. \tag{10.9}$$

These can be made into a two-equation DAE set, by equating the two right-hand sides and using one of the equations, conveniently the simpler, second one. This yields the DAE system

$$0 = V_f - V_b - K \sum_{i=0}^{n-1} \beta_i C_i'$$
$$\frac{d\theta}{dT} = K \sum_{i=0}^{n-1} \beta_i C_i' \,. \tag{10.10}$$

or, for the Langmuir isotherm, the nonlinear set

$$0 = K_f C_0 (1-\theta) - K_b \theta - K \sum_{i=0}^{n-1} \beta_i C_i'$$
$$\frac{d\theta}{dT} = K \sum_{i=0}^{n-1} \beta_i C_i' \,. \tag{10.11}$$

As for the transport- and isotherm-controlled case above, these equation sets can now be handled either using a standard implicit method or, perhaps logically in the case of a nonlinear isotherm, a Rosenbrock method.

11 Effects Due to Uncompensated Resistance and Capacitance

Electrochemists are aware of the annoying residual uncompensated solution resistance R_u between the Luggin probe and the working electrode, see for example [74]. Although it is possible in principle to compensate fully for the iR error thus introduced [131, 132], this is rarely done, as it introduces, in practice, undesirable instrumental oscillations or, in the case of damped feedback [132], sluggish potentiostat response.

The other often annoying fact electrochemists must live with is the double layer capacitance C_{dl}. This produces capacitive currents whenever the applied potential changes (see again [74]). The two effects work together, as capacitive currents also give rise to further iR errors.

With potential step methods, the capacitive current is a transient, decaying with a time constant equal to $R_u C_{dl}$. The usual procedure is to wait several of these time constants before making the current measurement, by which time the capacitive current has declined to a negligible value. It is therefore not a serious problem with potential step experiments.

Where both capacitive current and iR do interfere is with a.c. voltammetry (not gone into here) and LSV experiments. An early classic study is that of Nicholson [415], who investigated the effects of iR alone, pointing out that a simple correction, from measured currents and known R_u, for the potentials, does not work. The LSV curve becomes distorted and such a correction does not retrieve the shape of the curve as it would otherwise be in the absence of an iR effect. The reason is that the varying current during the sweep changes the electrode potential by a varying amount iR_u, and thus the potential program, that was intended to be linear with time, is no longer so. Bowyer et al. [128] and Strutwolf [529] show examples of such distorted potential-time relations and also distorted LSV curves, see also below.

The simulation literature deals with this problem sporadically, although it is often simply ignored. The iR effect introduces nonlinear boundary conditions (see below), and these have been dealt with in various ways. Gosser [274] advocates simple subtraction, using known measured currents of the experiment one is simulating in order to fit some parameter. Deng et al. [206] use a stepwise procedure that successively solves for each of the several unknowns without iteration. Iteration using binary searches have been used [162, 270, 529], as well as a Gauss-Seidel method [574]. Safford et al. [489]

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 193–199 (2005)
www.springerlink.com

rejected binary searching as too slow and Newton-Raphson iteration as unreliable, and used the van Wijngaarden-Dekker-Brent root-finding method, as described in Press et al. [452]. This is as reliable as a binary search (bisection) but faster, using a parabolic fit at each step. Despite the misgivings of some investigators, the best method is probably Newton-Raphson iteration, as used by Rudolph [477].

Simulations must thus handle the nonlinear boundary conditions. Some have taken the easy way out and used explicit methods [123, 429]. Bieniasz [105] used the Rosenbrock method (see Chap. 9), which makes sense because it effectively deals with nonlinearities without iterations at a given time step.

The classic work in this connection is that by Imbeaux and Savéant [313], who took the integral equation approach (see Chap. 9), incorporating the iR effects. They also established the formulation of the problem and the way to normalise both the uncompensated resistance R_u and double layer capacitance C_{dl}, adopted by most workers since then. Their normalisation of R_u followed that of Nicholson [415].

In the next discussion, only the LSV problem will be considered, since it is here that the major problems lie. The capacitive current component is, at any given time, given by

$$i_c = -C_{dl}\frac{dE}{dt} \tag{11.1}$$

where the negative sign is intended to produce a (positive) cathodic current from a cathodic-going sweep. This current will give rise to an iR error in the applied potential, equal to $+i_cR_u$ (that is, the applied potential will be a little more positive than intended). The Faradaic current will contribute a similar iR error.

First we must normalise some quantities, to make them compatible with the other dimensionless parameters already used. We refer to the normalisation formulae on p.26. Recall that we have normalised voltage by the factor $\frac{n\mathcal{F}}{\mathcal{RT}}$ and that the time unit τ for LSV is equal to $\frac{\mathcal{RT}}{n\mathcal{F}v}$ (v being the sweep rate), or the time the sweep takes to traverse one normalised potential unit p.

Resistance has units of volts per amperes, and thus must be converted to p units per G units. Using the normalisations in Chap. 2, this comes to

$$\rho = R_u\frac{n\mathcal{F}}{\mathcal{RT}}nFD^{\frac{1}{2}}c^*\sqrt{\frac{n\mathcal{F}v}{\mathcal{RT}}}\ . \tag{11.2}$$

This is as presented in [313], and is not normally simplified further. For capacity, which has units of current × time per volts, these become GT units per p units here and conversion leads to

$$\gamma_c = C_{dl}\frac{1}{n\mathcal{F}D^{\frac{1}{2}}c^*}\sqrt{\frac{\mathcal{RT}v}{n\mathcal{F}}}\ , \tag{11.3}$$

also normally written in this unsimplified form.

11.1 Boundary Conditions

It is solely in the boundary conditions that simulations differ from those without iR effects. We find that for a general electrochemical reaction (ignoring homogeneous reactions in this context), involving the two species A and B, and a set nominal potential p_{nom}, we have six boundary quantities and thus six equations for them. In fact, it is quite easy to reduce them to a set of four by elimination of two of the currents, but for the sake of clarity, they are left as formulated.

We have the following unknown boundary values: the two species' near-surface concentrations $C_{A,0}$ and $C_{B,0}$, the two species' fluxes, respectively G_A and G_B, the additional capacitive flux G_c, and the potential p, differing (for $\rho > 0$) from the nominal, desired potential p_{nom} that was set, for example, in an LSV sweep or a potential step experiment. Five of the six required equations are common to all types of experiments, but the sixth (here, the first one given below) depends on the reaction. That might be a reversible reaction, in which case a form of the Nernst equation must be invoked, or a quasi-reversible reaction, in which case the Butler-Volmer equation is used (see Chap. 6 for these). Let us now assume an LSV sweep, the case of most interest in this context. The unknowns are all written as future values with apostrophes, because they must, in what follows below, be distinguished from their present counterparts, all known.

The unknown capacitive flux G_c' is derived as follows. Equation (11.1) becomes, in dimensionless terms,

$$G_c' = -\gamma_c \frac{dp}{dT} \,. \tag{11.4}$$

Imbeaux and Savéant [313] provide the equation for the changed potential, which translates in present terms into the equation

$$p = p_1 - T + \rho(G_c + G_A) \tag{11.5}$$

(sweeping in the negative direction) and this gives, after differentiation,

$$\frac{dp}{dT} = -1 + \rho \left(\frac{dG_A}{dT} + \frac{dG_c}{dT} \right) \,, \tag{11.6}$$

that is, both the Faradaic and capacitive currents affect the potential if there is an iR drop. We can now construct all the needed boundary equations.

In the reversible case we have

$$C_{A,0}' - \exp(p')\, C_{B,0}' = 0 \tag{11.7}$$

as the first equation, with three unknowns (including p'). For a quasireversible system, the Butler-Volmer equation applies, instead,

$$G'_A = K_f\, C'_{A,0} - K_b\, C'_{B,0} \tag{11.8}$$

as described on page 92. We then have the flux equality equation

$$G'_A + G'_B = 0\ , \tag{11.9}$$

and the numerical approximation to the two fluxes

$$\begin{aligned} G'_A - \sum_{i=0}^{n-1} \beta_i C'_{A,i} &= 0 \\ G'_B - \sum_{i=0}^{n-1} \beta_i C'_{B,i} &= 0\ . \end{aligned} \tag{11.10}$$

The capacitive flux (11.4) is combined with (11.6) and the time derivatives are approximated by the two-point formula

$$\frac{dG}{dT} \approx \frac{G' - G}{\delta T}\ . \tag{11.11}$$

From the vantage point of time $T + \delta t$, these are backward differences. This gives

$$G'_c + \frac{\rho}{\delta T}\,(G'_c + G'_A) = \gamma_c + \frac{\rho}{\delta T}\,(G_c + G_A)\ . \tag{11.12}$$

Lastly, (11.5) is put into the form

$$p' - \rho(G'_c + G'_A) = p_1 - T' = p'_{nom}\ . \tag{11.13}$$

It will be noted that the equation pair (11.10) contains further unknowns $C'_{A,i}$ and $C'_{B,i}$, for $i > 0$. These can however be eliminated as described in Chap. 6, using the u-v mechanism. We assume that some implicit method is used here and that the first, backward, Thomas scan has been performed. Then, as described in that chapter, Sect. 6.2 or, for coupled systems, Sect. 6.4, concentrations can be expressed in the form

$$C'_i = u_i + v_i\, C'_0 \tag{11.14}$$

for both species. Equations (11.10) then become

$$\begin{aligned} G'_A - C'_{A,0} \sum_{i=0}^{n-1} \beta_i v_{A,i} &= \sum_{i=0}^{n-1} \beta_i u_{A,i} \\ G'_B - C'_{B,0} \sum_{i=0}^{n-1} \beta_i v_{B,i} &= \sum_{i=0}^{n-1} \beta_i u_{B,i}\ , \end{aligned} \tag{11.15}$$

now only containing concentrations at $X = 0$ or $i = 0$ as unknowns. For convenience, we rewrite these as

$$\begin{aligned} G'_A - V_A\, C'_{A,0} &= U_A \\ G'_B - V_B\, C'_{B,0} &= U_B \end{aligned} \tag{11.16}$$

with the four constants obvious from (11.15).

As mentioned above, the two unknown fluxes G'_A and G'_B appearing in the set can be eliminated by the application of the approximation pair (11.16), but it might be clearer not to do this and leave the full set of six unknowns as they are.

Essentially everything has now been given. The six-equation set must be solved numerically, and the Newton method works very well, requiring normally only 2–3 iterations at most, since the changes over a given time interval are relatively small. For this purpose the unknowns are gathered into the unknowns vector $\boldsymbol{X} \equiv [C'_{A,0}\ C'_{B,0}\ G'_A\ G'_B\ G'_c\ p']^T$. Further treatment is now confined to a concrete example.

11.1.1 An Example

The case of a reversible reaction is assumed, requiring (11.7). The set of six equations given above are written as the system

$$\boldsymbol{F}(\boldsymbol{X}) = 0 \tag{11.17}$$

or, detailed,

$$\begin{aligned} C'_{A,0} - \exp(p')C'_{B,0} &= 0 \\ V_A C'_{A,0} + G'_A - U_A &= 0 \\ V_B C'_{B,0} + G'_B - U_B &= 0 \\ G'_A + G'_B &= 0 \\ \rho\frac{G'_A}{\delta T} + \left(1 + \frac{\rho}{\delta T}\right) G'_c - \gamma_c - \rho\frac{(G_A + G_c)}{\delta T} &= 0 \\ -\rho G'_A - \rho G'_c + p' - p'_{nom} &= 0\,. \end{aligned} \tag{11.18}$$

The Newton-Raphson method will now be described very briefly. For a more detailed description, see, for example, Press et al. [452]. We assume that the present vector $\boldsymbol{X}$ is in error by a small amount $\delta\boldsymbol{X}$, and a short Taylor expansion leads to

$$F(\boldsymbol{X} + \delta\boldsymbol{X}) = F(\boldsymbol{X}) + \boldsymbol{J}(\delta\boldsymbol{X})\,. \tag{11.19}$$

$\boldsymbol{J}$ is the Jacobian of the system (11.18), that is, the derivative matrix, with respect to all the variables, of the system. It is

$$\boldsymbol{J} \equiv \begin{bmatrix} 1 & -\exp(p) & 0 & 0 & 0 & -\exp(p)C_{B,0} \\ V_A & 0 & 1 & 0 & 0 & 0 \\ 0 & V_B & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & \frac{\rho}{\delta T} & 0 & 1 + \frac{\rho}{\delta T} & 0 \\ 0 & 0 & -\rho & 0 & -\rho & 1 \end{bmatrix}. \tag{11.20}$$

The variables have been written without apostrophes. At the beginning of the Newton process, they have the old values, while the function $\boldsymbol{F}(\boldsymbol{X})$ supplies new constant values, driving the calculation. We demand that the corrected vector $\boldsymbol{X} + \delta\boldsymbol{X}$ satisfies (11.17), that is, that the left-hand side of (11.19) is zero. This leaves

$$\boldsymbol{J}\,\delta\boldsymbol{X} = -\boldsymbol{F}(\boldsymbol{X})\,, \tag{11.21}$$

a linear system that can be solved easily, for example using LUD decomposition.

Example program `LSV4IRC` (Appendix C) does this calculation. It is of interest to look at some results. The program was run with values $\rho = 1$ and $\gamma_c = 0.1$. Figure 11.1 shows all three fluxes. The dot-dashed (top) line is the total flux. Note that the LSV sweep goes from right to left. Note also the initial rise of the total current to the capacitive value, delayed by the time constant $\rho\gamma_c$. The solid line represents G_A alone, while the dashed (lowest at the left) line represents the capacitive current alone. In the absence of uncompensated resistance, this would rise to the constant value of 0.1 as soon as the sweep starts, because of the constant change in potential, and remain at that value. However, because of the iR effect, its rise is delayed, and its value does not remain constant as the faradaic current begins to rise, since then there are changes in potential beyond those due to the sweep, because of contributions by ρG_A to the potential. It is also of interest to see how much the faradaic current itself differs from that in the absence of iR effects. This is shown in Fig. 11.2. If one were to simulate only for G_A, one might conclude from that figure that the effect of iR is slight, but clearly, considering the total current shown in Fig. 11.1, it is not. It is this kind of curve one would

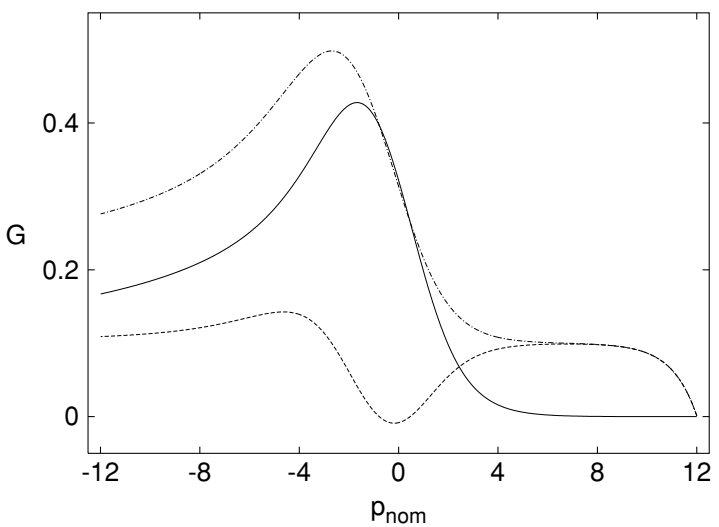


Fig. 11.1. LSV currents with uncompensated resistance and capacity

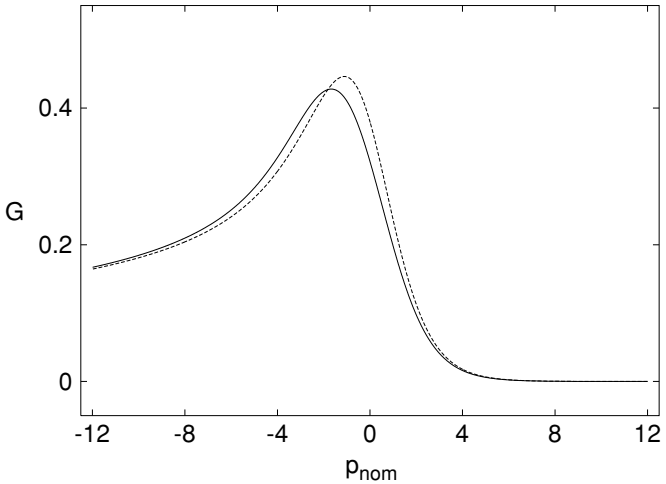


Fig. 11.2. LSV faradaic currents with and without uncompensated resistance

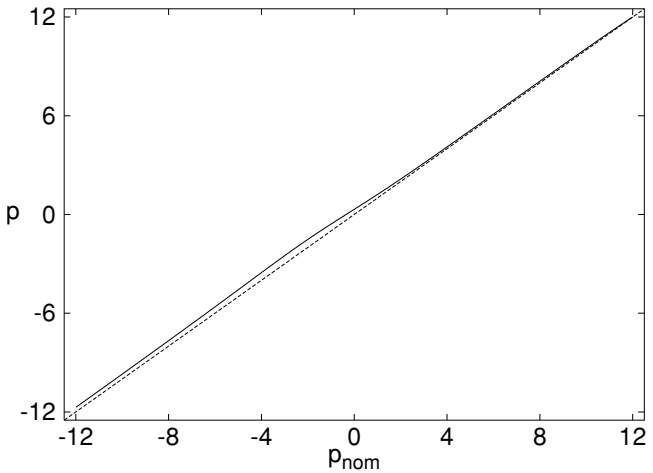


Fig. 11.3. LSV sweep potential with and without uncompensated resistance

obtain from an experiment, and would compare with a simulation. These marked changes arise from quite (visually) small deviations of the potential from the linear sweep, as seen in Fig. 11.3. Thus we can conclude that iR effects ought to be included in LSV simulations.

12 Two-Dimensional Systems

Electrochemical cells are of course three-dimensional. In preceding chapters, symmetry or a lack of concentration gradients in two of these dimensions has been assumed, thus conveniently reducing the system to one dimension. This is not always possible, and in fact in recent decades, some of the most popular electrodes require at least two dimensions for reasonable simulations. These are first and foremost the ultramicro electrodes (UMEs), in the various forms of a disk, band electrodes, and more recently the scanning electrochemical microelectrodes, as well as others. UMEs have also been assembled into arrays of such, increasing the difficulties. All of these fortunately have zero gradients in one of the three directions and thus require "only" two dimensions for their representation. Going from one to two dimensions, however, is a major step, requiring programming sophistication in order to avoid using too much computing time for a given simulation. In what follows here, the ultramicrodisk electrode (UMDE) will serve as the model for how to proceed, although the others are mentioned, and references and some theory are provided.

Until the 1960's, the dropping mercury electrode (DME) dominated electroanalytical chemistry (see such standard texts as [74, 257, 559] for details, and complications). It could be idealised as a sphere (disregarding the shielding due to the capillary) and had the advantages of a clean and smooth electrode surface, and a very wide potential working range, due to the high hydrogen overvoltage for water reduction at mercury. However, it necessitated the handling of liquid mercury, and it has mostly been replaced by the very small solid electrodes used today. One of the first to appear was the rotating disk electrode [362] which however is mentioned later in Chap. 13, as it involves convection. It then led to stationary disk electrodes, and the other stationary ones listed above.

Today, a number of ultramicroelectrodes are in use. They include the disk electrode, flat or hemispherical, the flat types being either inlaid – that is, flush with and embedded in an insulating plane – or recessed or protruding; band electrodes, either flat or hemicylindrical; and arrays of all these. They generally measure $<10\,\mu$ in one of their dimensions (diameter, width); hence the attribute "ultramicro-". The flat electrodes, flush with the insulating plane, have a problem of very large local current densities at the electrode edges. Bond et al. [125] mention this problem and cite Engstrom et al. [221]

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 201–233 (2005)
www.springerlink.com

for what they call an illuminating demonstration of the effect by experiment. The interesting paper by Zeiri et al. [581] also gives evidence for high edge current densities. The edge effect is responsible for the problems in simulating these electrodes. No matter how closely one packs grid lines near the edges, results are disappointing, and conformal mapping techniques are indicated.

The book by Fleischmann et al. [249] is a useful source of information on all the used UMEs, both for experimental and theoretical work. Recent reviews are useful, such as that of Aoki [57], Amatore [46] and Speiser [523], as well as the rather thorough section on microelectrodes in Bard and Faulkner [74] and the detailed review by Heinze [299] and [114], covering the period of 1996 to early 2004.

In this chapter, although the various microelectrode geometries are described and literature is provided both to theory and simulation work, the emphasis is laid on the UMDE. The principles of simulation at this electrode are then applicable to the others.

12.1 Theories

12.1.1 The Ultramicrodisk Electrode, UMDE

Initially, efforts were made to find expressions for the deviation of currents at the UMDE from that at a so-called planar electrode, which is the unidimensional case (called a "shrouded plane" by Oldham [425]). One can regard this as a disk (or any shape) at the bottom of a deep well, so that the system can be reduced to one dimension. Here the Cottrell equation defines the current for a potential step, as in Chap. 2 (2.37) and (2.44). At a flush UMDE, the current deviates from the Cottrell value very soon after the potential jump. Lingane [365] suggested that to a good approximation and for a range of small values of time t, the current i_{UMDE} at a UMDE could be expressed as

$$\frac{i_{UMDE}}{i_{Cott}} = 1 + A\left(\frac{Dt}{a^2}\right)^{\frac{1}{2}} \tag{12.1}$$

where a is the radius of the UMDE. He measured this experimentally, and found little deviation from the straight line for a range of t, and determined the slope A to be 2.12. In the same year, Soos and Lingane studied the problem mathematically [515] and found that the slope was 2.26 or $4/\sqrt{\pi}$; they considered this value a slight overestimate but in good agreement with experiment. This factor was then the subject of further study following these two papers. It was measured again and found to be 1.79 [326], then determined by simulation by the same team [327], again 1.79. Heinze [297] measured it also in his classic simulation study of the UMDE, and found it to lie in the range $1.77 \ldots 2.26$. Flanagan and Marcoux [246] found, by simulation, a value of 1.92. Shoup and Szabo [507] corrected the value of Kakihana et al. [327],

arriving at the correct one, $\sqrt{\pi}$ or 1.77. This is given again indirectly in the summary paper by Aoki [57], where other electrode geometries are also considered.

These attempts to express deviations of the current at a UMDE from the Cottrell current are somewhat fruitless because the expressions do not hold for other than rather small t values or rather, dimensionless values of the normalised time, Dt/a^2. General solutions were – and are – needed. There have been no analytical solutions holding for all times, but some limiting expressions, and an approximate one, have been derived.

Consider Fig. 12.1, depicting the UMDE in a cylindrical coordinate system. The electrode of radius a is flush with an infinite insulating plane. The *pde* that governs diffusion around the UMDE is then

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial r^2} + \frac{1}{r}\frac{\partial c}{\partial r} + \frac{\partial^2 c}{\partial z^2}\right) \tag{12.2}$$

with boundary conditions, for the "Cottrell" case (potential jump to a very negative potential), as follows:

$$\begin{aligned} t \le 0,\ \text{all } r, z: &\quad c = c^* \\ t > 0,\ r \le a,\ z = 0: &\quad c = 0 \\ r = 0: &\quad \frac{\partial c}{\partial r} = 0 \\ r > a,\ z = 0: &\quad \frac{\partial c}{\partial z} = 0 \\ r \to \infty,\ z \to \infty: &\quad c = c^* \ . \end{aligned} \tag{12.3}$$

For other kinds of experiments, the second, Cottrell condition, would be replaced with another.

For any given system, the current $i(t)$ at any one time is given by

$$i(t) = \int_{r=0}^{a} 2\pi \left.\frac{\partial c}{\partial z}\right|_{z=0} dr \ . \tag{12.4}$$

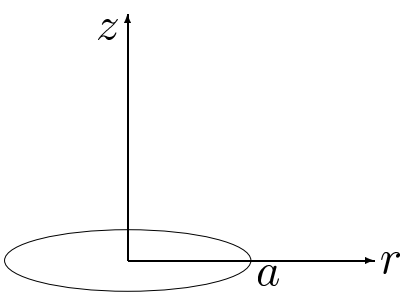


Fig. 12.1. Ultramicrodisk electrode coordinate system

The classic study of Saito (1968) [490] is often cited. Saito derived the steady state current i_{ss} at a UMDE, setting the left-hand side of (12.2) to zero, and arrived at

$$i_{ss} = 4nFc^*Da \ . \tag{12.5}$$

The steady-state value is the normalising quantity for the current as a function of time in most studies except those where the Cottrell current is used as the reference value. Saito also derived the concentration profile at steady state. It was printed incorrectly in the paper [490], but Crank and Furzeland [184] present the correct equation:

$$c = c^* \left(1 - \frac{2}{\pi} \sin^{-1} \left\{ \frac{2a}{\sqrt{z^2 + (a+r)^2} + \sqrt{z^2 + (a-r)^2}} \right\} \right) \tag{12.6}$$

for $z > 0$ and

$$c = c^* \left(1 - \frac{2}{\pi} \sin^{-1} \left\{ \frac{a}{r} \right\} \right) \tag{12.7}$$

for $z = 0, r > a$.

We want the current $i(t)$, a function of time. Aoki and Osteryoung [61] presented short- and long-time analytical expressions, the short-time one being the above (12.1), with $A = \sqrt{\pi} = 1.77$, that is

$$\frac{i_{UMDE}}{i_{Cott}} = 1 + \pi^{\frac{1}{2}} \left(\frac{Dt}{a^2} \right)^{\frac{1}{2}} \ . \tag{12.8}$$

This is better expressed directly without reference to the Cottrell current, and Aoki and Osteryoung [62] provide the short-time solution

$$i = i_{ss} \left(\frac{\sqrt{\pi}}{2\sqrt{\tau}} + \frac{\pi}{4} + 0.094\sqrt{\tau} \right) \tag{12.9}$$

where i_{ss} is the steady-state current as given in (12.5) and τ is the normalised time (more on that below). However, this approximation is not quite correct. Aoki and Osteryoung state [62] that they have adjusted the third coefficient (0.094) so that the approximation meshes better with their long-time approximation (see below). This was pointed out by Phillips and Jansons [442], who then presented the correct series:

$$i = i_{ss} \frac{\sqrt{\pi}}{4} \left(2\tau^{-\frac{1}{2}} + \pi^{\frac{1}{2}} + \tfrac{1}{4}\tau^{\frac{1}{2}} \right) \tag{12.10}$$

which makes the last coefficient in (12.9) 0.111 rather than 0.094. This slightly extends the range of applicability of the approximation.

For longer times, a complicated expression was derived by Aoki and Osteryoung [61], but it was incorrect, as pointed out by Shoup and Szabo [507],

who gave the correct expression, also given by Aoki in 1993 [57], with one more term. The first few terms of the long-time solution [57] are

$$I = i_{ss}\left(1 + 0.71835\tau^{-\frac{1}{2}} + 0.05626\tau^{-\frac{3}{2}} + 0.00646\tau^{-\frac{5}{2}}\ldots\right) . \qquad (12.11)$$

For very large τ, this becomes the steady state value of Saito (12.5). The steady state value might be considered the only exactly known expression, all others being approximations.

Shoup and Szabo also provide a general approximation that they state is accurate to 0.6% at all values of τ:

$$I = i_{ss}\left[0.7854 + 0.8862\tau^{-\frac{1}{2}} + 0.2146\exp(-0.7823\tau^{-\frac{1}{2}})\right] . \qquad (12.12)$$

A word is needed here about the definition of normalised time, in this context usually given the symbol τ. Most workers, including Shoup and Szabo, use the definition

$$\tau = \frac{4Dt}{a^2} . \qquad (12.13)$$

This is the definition assumed in the work of Shoup and Szabo [507] and Aoki and coworkers [57,61,62] and also by Gavaghan in some recent works [260,261]. The above three formulae (12.9, 12.11 and 12.12), are those for this definition of normalised time. One inconvenient side-effect of the definition is that, when one normalises the diffusion (12.2), using the new dimensionless variables' definitions

$$\begin{aligned} C &= c/c^* \\ R &= r/a \\ Z &= z/a \\ \tau &= 4Dt/a^2 , \end{aligned} \qquad (12.14)$$

the diffusion equation becomes

$$\frac{\partial C}{\partial \tau} = \frac{1}{4}\left(\frac{\partial^2 C}{\partial R^2} + \frac{1}{R}\frac{\partial C}{\partial R} + \frac{\partial^2 C}{\partial Z^2}\right) \qquad (12.15)$$

in which there is the factor 1/4. This is avoided by a different normalisation of time. Reverting now to the present context and using the symbol T (τ being here reserved for an observation time), we have the normalisation

$$T = \frac{Dt}{a^2} \qquad (12.16)$$

which eliminates the leading fraction. Using this definition, which the present author prefers (and which was also used by Flanagan and Marcoux [246]), the dimensionless diffusion equation is now as one expects,

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial R^2} + \frac{1}{R}\frac{\partial C}{\partial R} + \frac{\partial^2 C}{\partial Z^2} \,. \tag{12.17}$$

The normalised set of boundary conditions is then

$$\begin{aligned} T \le 0,\ \text{all } R, Z: &\quad C = 1 \\ T > 0,\ R \le 1,\ Z = 0: &\quad C = 0 \\ R = 0: &\quad \frac{\partial C}{\partial R} = 0 \\ R > 1,\ Z = 0: &\quad \frac{\partial C}{\partial Z} = 0 \\ R \to \infty,\ Z \to \infty: &\quad C = 1 \end{aligned} \tag{12.18}$$

and the current integration formula (12.4) becomes

$$I(T) = \int_{R=0}^{1} 2\pi \left.\frac{\partial C}{\partial Z}\right|_{Z=0} dR \,. \tag{12.19}$$

Also, the constants in the above three solution approximations change. They become the following new formulae: The short-time solution of Aoki and Osteryoung (12.9) is then

$$I = \left(\frac{1}{4}\left(\frac{\pi}{T}\right)^{\frac{1}{2}} + \frac{\pi}{4} + 0.188\, T^{\frac{1}{2}}\right) \tag{12.20}$$

(i_{ss} does not of course change); whereas the more correct formula as presented by Phillips and Jansons [442] is

$$I = \frac{\pi^{\frac{1}{2}}}{4}\left(T^{-\frac{1}{2}} + \pi^{\frac{1}{2}} + \tfrac{1}{2}\, T^{\frac{1}{2}}\right) \,. \tag{12.21}$$

The long-time solution (12.11) becomes

$$I = \left(1 + 0.35918\, T^{-\frac{1}{2}} + 0.007033\, T^{-\frac{3}{2}} + 0.000202\, T^{-\frac{5}{2}} \ldots\right) \tag{12.22}$$

and the general approximation (12.12) becomes

$$I = \left(0.7854 + 0.4431\, T^{-\frac{1}{2}} + 0.2146 \exp(-0.3912\, T^{-\frac{1}{2}})\right) \,. \tag{12.23}$$

There is, however, a much better pair of solutions, recently obtained by Mahon and Oldham [377]. They used what they call the "Cope-Tallman" method, involving the Green function, to find a much improved short-time and long-time solutions for the current at a disk electrode. Their formulae express currents at T values as defined above (12.13) (previously designated by τ), and normalised by $\pi nFDac^*$, rather than the steady-state value. Here

they are converted to the present scale by the simple expedient of a multiplication factor, and time is normalised to T by (12.16). The short-time approximation is then

$$I = \frac{\pi}{4}\left((\pi T)^{-\frac{1}{2}} + 1 + \frac{1}{2}\left(\frac{T}{\pi}\right)^{\frac{1}{2}} - 0.12003\,T + 0.013273\,T^{\frac{3}{2}}\right) \tag{12.24}$$

and their long-time approximation is

$$I = 1 + \frac{\pi}{4}\big(8\pi^{-\frac{5}{2}}\,T^{-\frac{1}{2}} + 8.9542\times10^{-3}\,T^{-\frac{3}{2}} - 2.5664\times10^{-4}\,T^{-\frac{5}{2}} - 2.2312\times10^{-4}\,T^{-\frac{7}{2}} + 2.7628\times10^{-5}\,T^{-\frac{9}{2}}\big)\,. \tag{12.25}$$

Unlike the approximations of Aoki and Osteryoung, these two actually overlap in their regions of applicability, as is seen in the next section.

Ranges of Applicability

From some simulations, in which reference current values were computed over a large range of times [151], it was possible to assess the range of applicability of the approximations. If we require that currents be accurate within 0.1%, then the short-time approximation of Aoki and Osteryoung, in its corrected form (12.21), is accurate in the range $0 \le T \le 0.03$, while their long-time expression (12.22) is applicable for $T > 0.5$. There is thus a gap in the range $0.03 < T < 0.5$, in which neither approximation yields good values. In the gap range, it was found that errors due to the approximations peaked at about 0.8%.

The universal approximation of Shoup and Szabo (12.23) has a similar, but wider gap range $0.002 \le T \le 10$, within which it has two excursions as high as 0.6% in amplitude. This is in accord with the original statement in the work [507], guaranteeing a maximum error of 0.6%.

Finally, the same study showed that the short-time solution of Mahon and Oldham (12.24) is 0.1% accurate in the range $0 \le T \le 1$, and the long-time approximation in the range $T \ge 0.4$. Thus, the two formulae yield accurate current values over the whole time scale without a gap.

LSV

For LSV, the diffusion equation for a UMDE is a little different from that for a potential jump. The LSV case can be considered as one of a group of possible cases, in which the characteristic time is defined independently of the disk radius. In general, let that characteristic time be τ. For LSV, as described on page 26, it is the time taken by the potential to sweep over one

dimensionless potential unit. There may be other definitions of τ. Then, time t is normalised by this τ, while distances (r, z) are normalised, as above, by the disk radius a. The time interval τ also leads to a Nernst diffusion layer thickness of $\sqrt{D\tau}$ (again, quite independent of disk radius a). From this follows an extra parameter in the diffusion equation, following the ideas of Heinze [298] and Aoki et al. [58] (both for LSV, but here more generally):

$$\frac{\partial C}{\partial \tau} = \frac{1}{P^2}\left(\frac{\partial^2 C}{\partial R^2} + \frac{1}{R}\frac{\partial C}{\partial R} + \frac{\partial^2 C}{\partial Z^2}\right) \tag{12.26}$$

with

$$P = \frac{a}{\sqrt{D\tau}} \tag{12.27}$$

that is, the ratio of disk radius to the Nernst diffusion layer thickness or, for LSV, where there is a relation between τ and the scan rate v, that is, $\tau = n\mathcal{F}v/\mathcal{RT}$,

$$P = (nFa^2 v/RTD)^{\frac{1}{2}} \ . \tag{12.28}$$

The symbol P is normally rendered as p, but this collides with our p for the dimensionless potential. For LSV, a small P value means a slow LSV sweep rate and a sigmoidal steady-state response, while a large value means a fast sweep rate, with the UMDE behaving more like a planar (shrouded) electrode. It will be seen later that the above impinges on the choice of a maximum Z and R value, which must be set such that they contain a sufficient number of $\sqrt{D\tau}$ units, according to the experiment. This will be discussed in some detail in the simulation section, below.

12.1.2 Other Microelectrodes

Theories for other microelectrodes are not as well developed as those for the UMDE but some approximations do exist. There are some reasonable approximations for the ultramicroband electrode. This is a relatively long strip, most often flush with the insulating plane it is embedded in. Since it is relatively long, that dimension can be ignored in the *pde*s describing transport at the electrode. The diffusion equation is very similar to that for the UMDE (12.2), removing the term in $r^{-1}\partial c/\partial r$, and if we call the half-width of the strip a, the normalisation is also very similar. There has been a series of theory papers on this electrode, a good example being that of Szabo et al. [543]. Their paper is interesting for another reason, the relationship between the flat band and hemicyclindrical electrodes, see below. For the microband electrode of half-width w and length l, Szabo et al. provide a short-time equation,

$$\frac{i(T)}{nFDc^* l} = \frac{1}{\pi T} + 1 \ , \tag{12.29}$$

with now $T = Dt/w^2$. This equation is said to hold to within 1.3% up to $T < \frac{2}{5}$. A longer-time solution, for higher values of T, also accurate to 1.3% is then

$$\frac{i(T)}{nFDc^*l} = \frac{\pi \exp(-2\sqrt{\pi T}/5)}{4\sqrt{\pi T}} + \frac{\pi}{\ln\left[(64\exp(-\gamma)T)^{\frac{1}{2}} + \exp(\frac{5}{3})\right]} . \quad (12.30)$$

They do present a long-time better approximation, but it is in Laplace-transformed form.

Coen and coworkers [167, 177, 180] have presented solutions to the diffusion-limited current at a band electrode in the form of integral equations; these must then be evaluated numerically, so this might be regarded as simulation.

Other microelectrodes are the hemispherical and hemicyclindrical geometries, long since understood (see Bard and Faulkner [74]), ultramicroring electrodes, which can be regarded as infinite bands [177], a conical microelectrode [585], a sphere-cap microelectrode [502], and arrays of all types of microelectrodes. A range of microring electrode thicknesses was considered by Amatore et al. [48], who found that a thick ring (that is, the ring width is comparable with the diameter) behaves more like a disk, while a thin ring approaches a band, as observed previously [177]. The Compton group has investigated what they call the shrouded ring system [156, 193, 194], as a model for a partially blocked electrode. Some theory has been attempted on arrays, especially band arrays, in the form of interdigitated electrodes, where anode and cathode bands alternate [56, 59, 63, 500], but results always need backing up with numerical calculations. There is much recent interest in the scanning electrochemical microscope (SECM), where no theory has been developed to date, and simulation is the rule here. The method was invented by Engstrom et al. [222], and one of the most recent papers on the subject is that of Mauzeroll et al. [390], where many good references can be found. A section on UMEs and their simulation in recent years (1996–2004) is found in the review of Bieniasz and Britz [114].

12.1.3 Some Relations

It is intuitively obvious that at longer times, when the diffusion layer thickness far exceeds the radius of a disk or hemisphere (for small P), or of the width of a band or the hemicylinder, currents at flat electrodes (disk, band) must resemble those at round electrodes (hemisphere, hemicylinder). Some relations between these have been established. Oldham found [427] that the steady-state currents at a microdisk and microhemisphere are the same if their diameters along the surfaces are the same. This means that for a microdisk of radius a, the steady-state current is the same as that at a microhemisphere of radius $2a/\pi$. At band or hemicyclindrical electrodes, there is

no steady state, but there still exists a relationship between them at long times. Szabo et al. [543], using the Laplace transform analytical solution for the current at a microband, and comparing it with that at a hemicyclinder, conjectured that the current at a band of width w has the same long-time current as at a hemicyclinder with radius $w/4$. This was borne out by simulated values.

12.2 Simulations

In the present context, we are interested in how best to simulate electrochemical processes at a two-dimensional electrode. The flat disk, the UMDE, is taken as the only example, as the techniques that have been developed for it are the same as those for the other geometries.

All the microelectrodes already mentioned have been simulated.

The UMDE evinces strong edge effects or very uneven current densities along the radius of the disk. It shares this problem with all the other flat microelectrodes such as microbands or rings. The exception of course are the hemispherical or hemicylindrical microelectrodes, which have no effective edges and behave as half of a sphere or cylinder, and also deeply recessed disks or bands, which approach the shrouded types.

It was early realised (for example by Crank and Furzeland in 1977 [184]) that the singularity at such edges will degrade overall accuracy in a simulation. In fact, Gavaghan points out [259, 260] that because of this effect, the simulation error in calculated concentrations is of $O(h^{1/2})$, h being the interval size in space, near the edge. This is rather poor, so that unless special techniques (see below) are used, these simulations can be very cpu-intensive. The worst methods for simulating such systems are the explicit method and equal intervals in (r, z) space (for the UMDE); nevertheless, both have been used. Motivation for using simple explicit or splitting methods is that the more efficient implicit methods are not easy to implement in two dimensions. For this reason, hopscotch [275] and ADI [436] (see below) have been favourites in this area, as they obviate the need for solving largish banded systems of equations, as one has with the true implicit methods. However, more recent work has indeed made use of implicit methods, combined with sparse matrix techniques, as described below.

Flanagan and Marcoux [246] were the first to attempt a UMDE time-marching simulation, in order to find the constant in the approximation of Lingane's (12.1); they used the explicit method. Crank and Furzeland [184] addressed the steady state for the UMDE and described some of the problems; they also briefly mention time-marching simulations. Their work appears to have come just after that of Evans and Gourlay [224], who used hopscotch. They also found some oscillatory behaviour of the solution, which is not always mentioned. As Gourlay realised [275], hopscotch is mathematically

related to ADI, which in turn approximates Crank-Nicolson, known to be oscillatory in response to initial discontinuities such as a potential jump (more on this problem below).

Heinze is usually cited as the first to do a thorough study of the UMDE simulation [297], using ADI for the potential step problem. He followed this with an LSV study [298], and used unequal intervals in the next study [300], to come to grips with the edge effect. Meanwhile, Shoup and Szabo [507] applied hopscotch to the problem, and this method has continued to be used up to the present [52,394,488,489], also with other microelectrodes. As mentioned in Chap. 9, hopscotch has a problem, called "propagational inadequacy" by Feldberg [232]. Hopscotch becomes inaccurate for large time intervals, which are one reason for using stable algorithms. As well, as mentioned above, hopscotch does give rise to some initial oscillations for potential step simulations. These are less severe for small time intervals, however, and perhaps for that reason are not always mentioned or considered serious. Safford and Weaver [488] addressed the nontrivial problem of uncompensated resistance and double layer capacity in these simulations. Some continued to use the plain explicit method, despite the advantages of implicit or semi-implicit methods [557]. ADI, as mentioned, was used on some occasions [297,545]. Finally, Crank-Nicolson remains an attractive method, if sparse matrix solvers are used, especially if its oscillatory response can be damped. This can be done by some rather simple expedients, as described [149,432], such as either starting a potential jump simulation by subdividing the first step in time into sub-intervals, or – even simpler – to make the very first 1–4 steps BI steps. BI is known to have a very steady response to initial transients, and it turns out that after 1–4 steps, when CN is resumed, no more oscillations are seen. What is more, this was especially effective with UMDE simulations in conformal space.

It was soon realised that at least unequal intervals, crowded closely around the UMDE edge, might help with accuracy, and Heinze was the first to use these in 1986 [300], as well as Bard and coworkers [71] in the same year. Taylor followed in 1990 [545]. Real Crank-Nicolson was used in 1996 [138], in a "brute force" manner, meaning that the linear system was simply solved by LU decomposition, ignoring the sparse nature of the system. More on this below. The ultimate unequal intervals technique is adaptive FEM, and this too has been tried, beginning with Nann [407] and Nann and Heinze [408, 409], and followed more recently by a series of papers by Harriman et al. [287,288,289, 290, 291, 292, 293], some of which studies concern microband electrodes and recessed UMDEs. One might think that FEM would make possible the use of very few sample points in the simulation space; however, as an example, Harriman et al. [292] used up to about 2000 nodes in their work. This is similar to the number of points one needs to use with conformal mapping and multi-point approximations in finite difference methods, for similar accuracy.

In general, the finding of Rudolph [478], that in one-dimensional simulations, direct discretisation on an unequally spaced grid, rather than equal spacing on a transformed grid, is best, does not appear to apply to UME simulations. Gavaghan made a very thorough study of UMDE simulations [259,260,261,262] and concluded that the above-mentioned $O(h^{\frac{1}{2}})$ behaviour limits the convergence obtained. Much better are transformations by conformal mapping, to eliminate the edge singularity. Such conformal maps were used as early as 1966 (Newman, in a mathematical study of the rotating disk [411]) and Saito [490], who worked out the steady-state current at a UMDE, used conformal mapping; in fact he used the same formula as later applied by Michael et al. [394]. Safford et al. [489] used the same formula, and more is said about this technique below.

Other UMEs have been simulated, and are briefly mentioned here. Microband or microhemicyclinders were simulated starting in 1986 (Deakin et al. [197], Amatore et al. [51]), mostly using hopscotch. Coen et al. [167], followed by Cope et al. [177, 180] used the integral equation method (see Chap. 9) to simulate microband or microrings. Jin and coworkers used their FAM method on microbands and -rings, as well as on a microoblate spheroidal electrode [316, 317, 319, 454, 455, 456]. Varco Shea and Bard [556] used the explicit method on microband arrays, Bieniasz and Britz [113] simulated chronopotentiometry at a microband using a Rosenbrock method; interdigitated microband electrodes (IDAs) were simulated [60, 316, 536]. There is recent interest in double microband or -hemicylinder pairs, in generator-collector mode, and some simulations have been described [47, 49, 65, 541], using explicit [65] and ADI [47, 49, 541] methods. Also, the scanning electrochemical microscope (SECM) with its similarity to a UMDE with a close TLC-like opposite wall and its various boundary conditions there, demands simulation, and some work along these lines has been done [53, 75, 76, 77, 78, 204, 237, 375, 383, 384, 390, 503, 512, 530]. This list is probably not exhaustive.

12.3 Simulating the UMDE

The ways to simulate our chosen example, the UMDE, are described here. The integral equation approach, taken by Coen and coworkers over a number of years [167, 176, 177, 178, 179, 180, 219] for microband electrodes, can be used on the UMDE as well [179]. The reader is referred to these papers for the method. Also, although the adaptive FEM approach might be thought to be about the most efficient, and has been developed by a few workers (see above, references to Nann and Heinze, and Harriman et al), it does not seem the most popular; it is not trivial to program, and as Harriman et al. found, it appears that a rather large number of nodes were required. The reason is probably that this is a kind of discretisation in the original cylindrical (R, Z)

space, where convergence, as mentioned above, is of $O(h^{\frac{1}{2}})$ [259], and many nodes are needed to get reasonable results. This is of course also the case with finite differences, as described below. Discussion here is confined to the use of finite difference methods for UMDE simulation, since these serve as guides for the simulation of the other 2D electrodes.

One has the choice between applying finite difference discretisation either directly to a grid of points in the cylindrical (R, Z) space, or to a transformed space. In one dimension, it has been found [478] that direct discretisation without transformation is better. In the case of 2D simulations where edge effects are seen, this is not the case, and transformation is better. Both approaches are described here.

12.3.1 Direct Discretisation

The discussion to follow refers to the example program `UMDE_DIRECT` (Appendix C). Consider Fig. 12.2, which is the normalised version of Fig. 12.1, the disk edge now lying at $R = 1$. When discretising directly here, we must decide on maximum values for R and Z, as indicated, see below. Clearly, given the poor convergence, many grid points are needed, which in turn points to the use of unequal point spacing in both axes. This was done by Gavaghan in his three-part study [260,261,262], in which he employed a scheme similar to that pictured in Fig. 12.3.

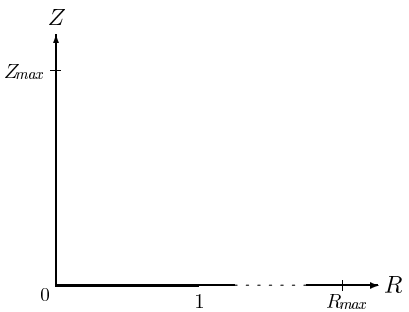


Fig. 12.2. Coordinate system for the UMDE

There is a slight complication in the setting of the maximum R and Z values. The procedure depends on whether (12.17) or (12.26) is simulated. In the former case, we have

$$Z_{max} = 6\sqrt{T_{max}} \tag{12.31}$$

where T_{max} is the number of time units over which the experiment runs. These units are defined by (12.16) for this case, making the Nernst diffusion layer thickness equal to the UMDE radius for one time unit. For R_{max}, one should probably use the formula

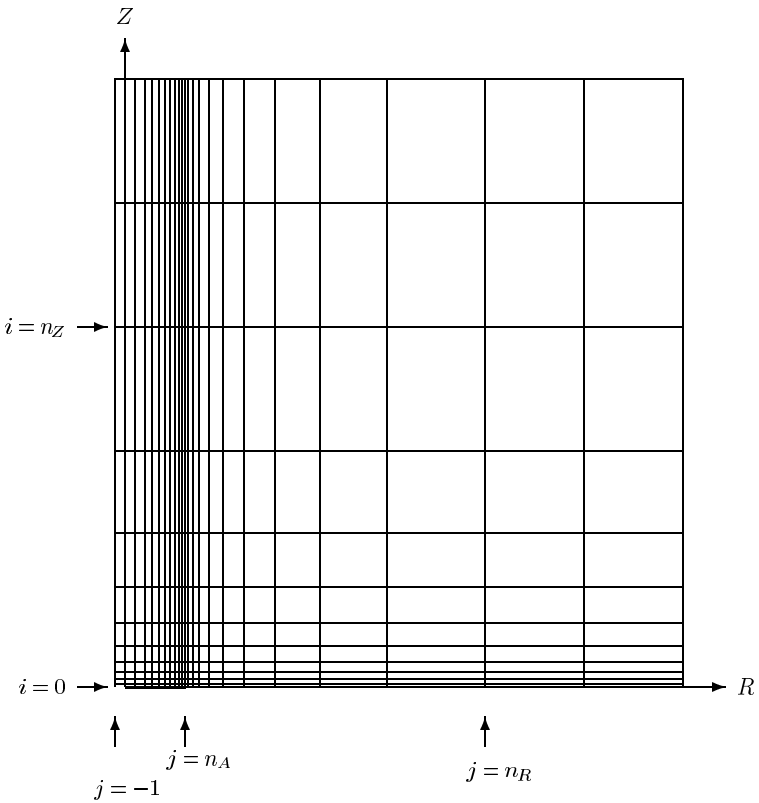


Fig. 12.3. UMDE unequally spaced grid in $[R, Z)$ coordinates

$$R_{max} = 1 + 6\sqrt{T_{max}}\,, \tag{12.32}$$

taking into account that the extent is measured from the disk edge, one disk radius from the origin.

If the characteristic time is defined independently of the disk radius, and diffusion (12.26) results, the Nernst diffusion layer thickness is dependent only on the number of these time units. So if the characteristic time is τ and the maximum duration of the experiment is τ_{max} (giving $T_{max} = \tau_{max}/\tau$), then the final diffusion layer thickness is $\sqrt{D\tau_{max}}$. Then, in dimensionless distance units (normalisation being division by the disk radius a), this becomes, after multiplying by 6 and noting (12.27),

$$Z_{max} = \frac{6}{P}\sqrt{T_{max}} \tag{12.33}$$

and for R,

$$R_{max} = 1 + \frac{6}{P}\sqrt{T_{max}}\,. \tag{12.34}$$

This will become a little more complicated later, when the space is mapped into new coordinates, and limits in terms of these must be set. In Fig. 12.3, the number of nodes (lines) is held small, in order not to confuse the picture. The grid is chosen such that there are expanding intervals in both the Z direction and in the two R directions away from and on either side of the line $R = 1$. For reasons which will become clear below, the index j for the R positions starts at -1. There are n_A intervals between $R = 0$ and the disk edge, and a total of n_R between the origin and the point at which $R = R_{max}$; there are two further points beyond this, which also will be explained. The positions for Z, indexed with i, begin at zero and n_Z is the point at which $Z = Z_{max}$. Again there are two further points beyond this value.

The unequal grid was generated using the Fortran function `EE_FAC` (Appendix C and described in Sect. 7.2). One needs to decide the numbers of points in the three ranges, and the minimum intervals, whereupon `EE_FAC` produces the required γ values for the expansion. One range goes from $R = 1$ backwards to $R = 0$, and we have a further point indexed $j = -1$, as seen in Fig. 12.3; this point is chosen such that its distance from $R = 0$ is the same as that of the first point to the right of $R = 0$. In the other direction, the expansion finds the final variable point at R_{max}, and there are two further points beyond it. These do not need further expansion, and the two intervals are made the same as the final one reaching out to R_{max}; similarly for the Z axis. Thus the last four outermost points in both the R and the Z directions are equally spaced. This is somewhat arbitrary, and the outermost points could be placed even closer to the last variable points (possibly) to improve the discretisation accuracy.

The reason for there being two extra points at the outer boundaries is the fact that the second derivative approximation on an unequal grid is only first-order with respect to the intervals (see Chap. 3, Sect. 3.8 on page 44), if three-point approximations are used. Gavaghan [260] and Verbrugge and Baker [557] write incorrectly that these approximations are second-order. In a private communication (2003), Gavaghan explained to the present author that while this is strictly not true, the nature of the exponentially expanding grid makes the multiplication constants rather small, so that second-order-like accuracy is achieved. Nevertheless, when one measures the order, it is clearly first order. For this reason, it was decided in a study [532] using direct discretisation to use asymmetric four-point formulae, which are second-order for the second derivative on an unequal grid. This is the reason why two extra points beyond the limit points were set in the grid, as the approximations were of the form $u_2''(4)$ in the terminology used in Appendix A.

We are now ready to apply the discretisations, but must decide on the vector of unknown concentrations at all the grid points in Fig. 12.3. It is convenient to include even the boundary points (but not those at $j = -1$, which serve only as fictitious points), setting these to known values in the large linear system to be generated. Thus we note that the total number N of unknowns is given by

$$N = (n_R + 3)\,(n_Z + 3)\ . \tag{12.35}$$

A convenient ordering of the grid points is achieved by arranging the concentration grid one row after the previous. The numbering of the elements in the vector is then the following. Index k of the vector element corresponding to the grid value C_{ij}, $(i = 0 \ldots n_Z + 2, \quad j = 0 \ldots n_R + 2)$ is

$$k = i\,(n_R + 3) + j + 1\ , \tag{12.36}$$

so that the whole grid is now mapped into N elements. The map is conveniently generated in the example program `UMDE_DIRECT` by the function `KMAP`. There is a corresponding function `UNMAP` which calculates i and j from a given k, needed in order to fold newly calculated concentration values back into the grid.

With N elements as the unknown vector, an $N \times N$ matrix is clearly required for the solution of the linear system of discrete equations. This can be rather large. One approach, that has been tried [138], is to ignore this problem and to actually generate a huge matrix and let the system be solved by a suitable solver. This limits the size of N, however, leading to somewhat inaccurate simulations, unless rather high-order approximations are used. However, when discretising, one notes a large number of zero elements in the matrix, which is banded, and this suggests a sparse matrix technique.

The program package **MA28** [215] was found to be useful. The package can be downloaded from the Harwell site [1]. It is written in `Fortran IV`, but there is no problem in adapting it to `Fortran 90`, thanks to the (so far) downward compatibility of the latest language definition. `MA28` does an LU-decomposition of a sparse matrix, allowing efficient solution by back-substitution after the initial LU-decomposition. What is more, for those cases where the matrix varies with time – as is the case in, for example, second-order homogeneous chemical reactions and in time-varying boundary conditions – `MA28` has the very convenient feature that it preserves some information from the first LU-decomposition and, as long as the sparsity pattern of the matrix does not change, subsequent LU-decompositions can be done much faster than the first. This package, then, was used in the program `UMDE_DIRECT`. Another package sometimes used is **Y12M**, available from `netlib` [2], and described in [584]. It offers the same features as MA28.

Firstly, the discretisation itself is described. We restrict the discussion to the BI time integration, in order to focus on the spatial discretisations. The program `UMDE_DIRECT` in fact uses BI as the first step, then three-point BDF, which produces second-order accuracy with respect to δT, this being the rational BDF startup described in Chap. 4, page 59. Take a point away from the boundaries, indices i (for Z) and j (for R). The discretisation at concentration $C_{i,j}$ of the *pde* (12.17) has three derivative terms, all to be discretised using four-point formulas. The coefficients can be precalculated. For the row along Z, there are, for each $0 < Z \leq Z_{max}$, that is, $0 < i \leq n_Z$, four coefficients for the approximations

$$\frac{\partial^2 C}{\partial Z^2} \approx \alpha_{Z1} C_{i-1,j} + \alpha_{Z2} C_{i,j} + \alpha_{Z3} C_{i+1,j} + \alpha_{Z4} C_{i+2,j} \tag{12.37}$$

(the coefficients can be fetched from the routine `U_DERIV`). These coefficients are independent of R (or j), so there are only $4n_Z$ of these. Similarly, there are $4n_R$ coefficients $\alpha_{Rk}, k = 1 \ldots 4$, for the approximations

$$\frac{\partial^2 C}{\partial R^2} \approx \alpha_{R1} C_{i,j-1} + \alpha_{R2} C_{i,j} + \alpha_{R3} C_{i,j+1} + \alpha_{R4} C_{i,j+2} \; . \tag{12.38}$$

This leaves the last term. For the moment, assume that we are away from the problem area $R = 0$, and we thus have the simple approximation with the last set of coefficients for the first derivative,

$$\frac{1}{R}\frac{\partial C}{\partial R} \approx \frac{1}{R_j} \left(\beta_{R1} C_{i,j-1} + \beta_{R2} C_{i,j} + \beta_{R3} C_{i,j+1} + \beta_{R4} C_{i,j+2} \right) \; . \tag{12.39}$$

These three formulae work for (almost) the whole field of values that undergo diffusional changes, up to the boundary lines, because we have made sure that there are enough points in the grid beyond these lines for the discretisation formulae to refer to. There is a problem area, as mentioned above, at $R = 0$, where the above approximation cannot be used, due to the singularity. This has been addressed by Crank and Furzeland [184] and again by Gavaghan [260]. The method they used is also described in detail by Smith [514]. It is the following. Expand $(\partial C/\partial R)$ at some small R, using Maclaurin's expansion (a special case of Taylor's expansion):

$$\frac{\partial C}{\partial R}(R) = \frac{\partial C}{\partial R}(0) + R\frac{\partial^2 C}{\partial R^2}(0) + \ldots \tag{12.40}$$

and, from boundary conditions (12.18), $\frac{\partial C}{\partial R}(0) = 0$, and letting $R \to 0$, we obtain

$$\frac{1}{R}\frac{\partial C}{\partial R}(0) \approx \frac{\partial^2 C}{\partial R^2}(0) \; . \tag{12.41}$$

Thus, we can simply add this term to the existing one, and the *pde* on the axis becomes

$$\frac{\partial C}{\partial T} = \left(2\frac{\partial^2 C}{\partial R^2} + \frac{\partial^2 C}{\partial Z^2} \right) \tag{12.42}$$

for which discretisations already exist (12.37) and (12.38). The only (small) problem is that in this case, $j = 0$, and the discretisation thus refers to points with index –1. This is easily overcome, again using the boundary condition $\frac{\partial C}{\partial R}(0) = 0$, which means that $C_{i,-1} = C_{i,1}$ and thus

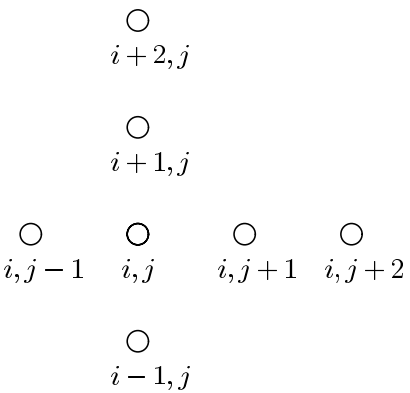
$$\frac{\partial^2 C}{\partial R^2}(R = 0) \approx \alpha_{R2} C_{i,0} + (\alpha_{R1} + \alpha_{R3}) C_{i,1} + \alpha_{R4} C_{i,j+2} \; . \tag{12.43}$$

It is now seen why the grid diagram contains the column at $j = -1$; that value of R is needed in order to get the four coefficients from `U_DERIV` at $R = 0$. A concentration value on this column need never be referred to.

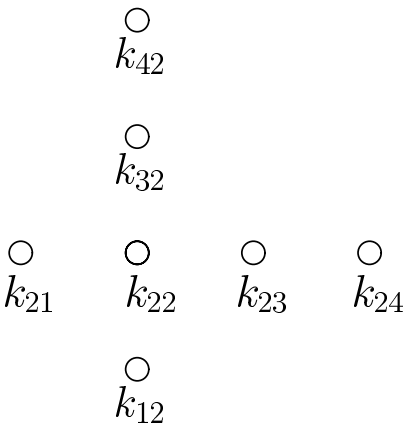
Another special area is the insulating plane outside the disk, defined by $Z = 0, R > 1$. Here, the boundary condition is usually given as in the set (12.18), zero gradient with respect to Z. This is expressed as a four-point first derivative, as

$$\beta_1 C_{0,j} + \beta_2 C_{1,j} + \beta_3 C_{2,j} + \beta_4 C_{3,j} = 0 \,. \tag{12.44}$$

To make the discretisation process more visual, consider any position (i, j) in the grid. There are a total of 7 points around and including this central point, and each of them has its own k-value, mapped from its indices. It looks like this:



Each of the positions maps into a k value, the index of the element in the unknowns vector to be solved for. These are denoted, corresponding to the above scheme, by



Thus, the central point at (i, j) has map-index k_{22}. For its discretisation, there will be entries in row k_{22}, at column positions at all seven k values. The horizontal row (referring to the mapping formula (12.36)) are all contiguous k values, while the vertical row maps into column values that are $n_R + 3$ apart from each other. So only k_{22} need be computed by the mapping function `KMAP`, the others can then be simply set: for example,

$$\begin{aligned} k_{21} &= k_{22} - 1 \\ k_{23} &= k_{22} + 1 \\ k_{12} &= k_{22} - n_R - 3 \end{aligned} \tag{12.45}$$

etc.

We can now put the discretisations together, still focussing only on the right-hand side of (12.17). Adding up the individual discretisations (12.37), (12.38) and (12.39), we can express the total (semi) discretisation as

$$\begin{aligned} \frac{dC}{dT} &= a_{21}C_{i,j-1} + a_{22}C_{i,j} + a_{23}C_{i,j+1} + a_{24}C_{i,j+2} \\ &\quad + a_{12}C_{i-1,j} + a_{32}C_{i+1,j} + a_{42}C_{i+2,j} \ , \end{aligned} \tag{12.46}$$

where the a-coefficients are put together as follows:

$$\begin{aligned} a_{21} &= \alpha_{R1} \\ a_{22} &= \alpha_{R2} + \alpha_{Z2} \\ a_{23} &= \alpha_{R3} \\ a_{24} &= \alpha_{R4} \end{aligned} \tag{12.47}$$

$$\begin{aligned} a_{12} &= \alpha_{Z1} \\ a_{32} &= \alpha_{Z3} \\ a_{42} &= \alpha_{Z4} \end{aligned} \tag{12.48}$$

with the α coefficients already defined above. For the axis where $R = j = 0$, the term in $C_{i,j-1}$ drops out.

This leaves the boundary conditions. The equation for the insulating plane is given in (12.44), producing four matrix entries. The remaining points are now those on the disk surface itself, and the points outside the diffusion space. On the disk surface, for the Cottrell-like simulation, we have zero concentrations, and at the outer points all concentrations are unity. These produce single entries in the matrix.

We must now attend to the time integration, that is, the choice of discretisation of the left-hand side of (12.17). In the example program `UMDE_DIRECT` it was decided to use a second-order time integration, and not CN. This suggested either extrapolation or BDF, both described in Chap. 8, Sect. 8.5.2. Second-order extrapolation has the disadvantage of requiring two half-sized steps in time as well as one whole step, which means two different coefficient

matrices and thus two LU-decompositions. This takes up more computer memory and cpu time. BDF, on the other hand, is done in a single step and requires, for its second-order variant, only a second concentration array, which is much smaller than the coefficient matrix. This matrix (in both cases) is smaller than it might be if we did not specify it as a sparse matrix for the routines in the `MA28` package, but is still rather large, compared to the concentration grid. BDF does have the start-up problem, and it was decided to use the rational start by taking a single BI step, followed by second-order BDF. This produces second order accuracy with respect to δT, and is quite stable.

For the BI step, the left-hand side of (12.46) is

$$\frac{dC'}{dT} \approx \frac{C'_{i,j} - C_{i,j}}{\delta T} \tag{12.49}$$

which, when putting unknowns and knowns on opposite sides, makes (12.46)

$$a_{21}C'_{i,j-1} + (a_{22} - \frac{1}{\delta T})C'_{i,j} + a_{23}C'_{i,j+1} + a_{24}C'_{i,j+2} + a_{12}C'_{i-1,j} + a_{32}C'_{i+1,j} + a_{42}C'_{i+2,j} = -\frac{C_{i,j}}{\delta T} \; . \tag{12.50}$$

For second-order BDF,

$$\frac{dC}{dT} \approx \frac{'C_{i,j} - 4C_{i,j} + 3C'_{i,j}}{2\delta T} \tag{12.51}$$

and the final lumped equation is

$$a_{21}C'_{i,j-1} + (a_{22} - \frac{3}{2\delta T})C'_{i,j} + a_{23}C'_{i,j+1} + a_{24}C'_{i,j+2} + a_{12}C'_{i-1,j} + a_{32}C'_{i+1,j} + a_{42}C'_{i+2,j} = \frac{'C_{i,j}}{2\delta T} - \frac{2C_{i,j}}{\delta T} \; . \tag{12.52}$$

There is only a small difference between the two forms and in the program, there is need for only a small `IF`-statement split, to handle both in the same code stretch. In fact, in `UMDE_DIRECT` the BI form is first assumed, and then, if BDF is found to hold, the change is made from one to the other.

A small problem is the current integration. This is defined, in dimensionless terms, in (12.19). The integration must be performed on an unevenly spaced set of R values. Gavaghan has examined current integration. From some numerical experiments on an evenly spaced grid [259] he concluded that the trapezium method is the most suitable. Simpson integration does not produce better results, because the edge anomaly produces large errors that dominate the current integration process. In his 1998 paper [260], describing a UMDE simulation on an unevenly spaced grid, Gavaghan then again opted for the trapezium method, and the three-point formula for evaluating the

flux densities $\partial C/\partial Z$ at each R and $Z = 0$. In our own numerical experiments, a Simpson-like algorithm was designed for unevenly spaced points, but while this worked fairly well for exact concentrations (those at the steady state), the trapezium method was just as good for simulated concentrations, agreeing with Gavaghan. However, four-point current approximations were used.

The resulting code can be seen in the example program `UMDE_DIRECT` which the interested reader might study. It turns out that a rather large number of points is required, and very small intervals near the disk and around the disk edge. To get good accuracy from this program, for example, it was necessary to use the maximum possible number of points on our computer, $n_A = n_Z = 180, n_R = 240$, and set the smallest $\delta Z = 10^{-6}$ and equally, the smallest $\delta R = 10^{-6}$, similar to the values chosen by Gavaghan [261]. The expansion parameters γ are then computed automatically to fit these numbers. With these parameters, the program produces currents with better than 0.1% accuracy for $T < 5$.

12.3.2 Discretisation in the Mapped Space

The other major approach to simulating an UME is to map the 2D-space into another space using conformal mapping. Consider Fig. 12.4, showing two sets of equiconcentration lines for a potential jump at a UMDE at $T = 1$. The lines range from zero (along the electrode in A, or along $\Gamma = 0$ for B) in steps of 0.1 up to 0.9. Note the crowding of the lines in A around the electrode edge, but the rather even spread of the lines in B, and the fact that they are almost parallel with the base line. The mapping function used in this case is that of Amatore and Fosset [52], about which more will be said below. The figure indicates that simulation in the transformed space should be better than in the original (R, Z) space, and this is indeed true, especially for larger values of T. However, consider now Fig. 12.5, the same situation but at $T = 0.01$. Here, the lines in normal space are, over most of the R-range, almost parallel with the base line except for a small area around the disk edge, while in transformed space, they are no longer parallel and somewhat crowded at the

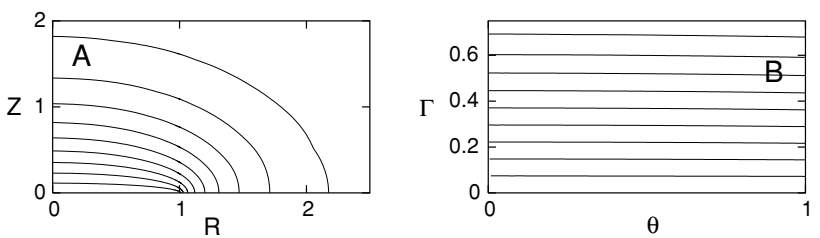


Fig. 12.4. Equiconcentration lines at T = 1 at a UMDE in normal and AF-transformed space

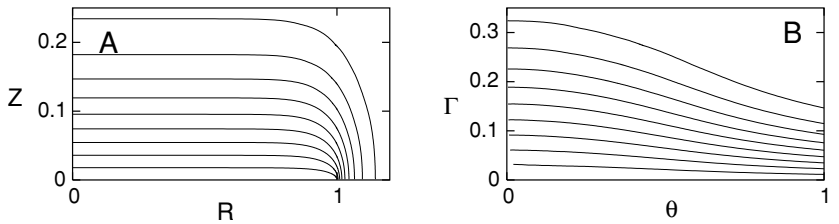


Fig. 12.5. Equiconcentration lines at T = 0.01 at a UMDE in normal and AF-transformed space

right-hand end (which corresponds to the the region near the central axis, $R = 0$). This suggests that direct discretisation in (R, Z) space might be favourable. In practice, however [532], conformal mapping is superior over the whole time range.

There is thus good argument for using transformation for 2D simulations. Several transformation formulas have been suggested and used. The first to do this was Newman in 1966 [411], for his study of the resistance to a flat disk; he was followed by Saito [490], in his derivation of the steady-state currents at a microdisk and microband electrode. Saito used a conformal mapping function for the band electrode, that was later used again by Michael et al. [394], applying it to the UMDE (see below). Amatore describes several conformal mappings in his review of UMEs [46]. The properties of conformal maps can be found in such publications as [524], and Amatore has a good discussion of the technique [46].

Some Transformations

Here, the four major mapping functions for the disk electrode are presented, as well as the form that the diffusion equation for the disk electrode takes in the mapped spaces. We assume that the cylindrical coordinates, time and concentrations have all been normalised by the disk radius as in (12.14).

Michael et al. [394] used the mapping function used earlier by Saito [490], transforming to elliptic coordinates [404],

$$\begin{aligned} R &= \cos\theta \cosh \Gamma \\ Z &= \sin\theta \sinh \Gamma \,. \end{aligned} \tag{12.53}$$

This will be called **MWA** here. This transformation is also used for band electrodes, with R replaced by X, measured as a distance from the centre of the band, across the band [46]. It results, in the case of the disk electrode, in a new diffusion equation, whose form is deferred to a later place, below.

We wish to simulate by discretising on an equally spaced grid in the transformed space, and this grid should place points optimally in the original (R, Z) space. That is, they should be closely spaced near the disk edge,

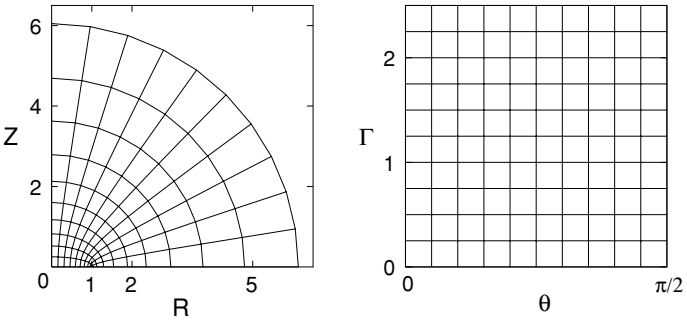


Fig. 12.6. A 10×10 grid in MWA (θ, Γ) space and its equivalent in (R, Z)

and more widely spaced, the further away from the edge point they are. For illustration, note Fig. 12.6, where a coarse 10×10 grid is shown in the transformed (θ, Γ) space of the transformation (12.53). The Γ end is open. This means that for a given simulation, one must decide on the maximum Γ value (see page 229). This may or may not be regarded as a disadvantage. This grid of points, retransformed by application of (12.53), produces the grid on the left-hand side of the figure. We note that indeed, points are closely spaced around the disk edge, and move apart away from that region. The outer, nearly circular curve corresponds to the maximum Γ chosen in this example, 2.5. Γ is the parameter that sets distance from the origin, in a slightly complicated way. We note that the regular grid of Fig. 12.6 is reflected in a rather regular spacing of the "radial" lines (angles) and an expanding spacing in the distance of the circle-like lines. All this is favourable, as it will tend to space isoconcentration lines at roughly equal intervals.

The MWA map has what might be regarded as a drawback. We wish to contain the concentration field that varies during the time T_{max} of the simulation, that is, to have distances of about $6 \times \sqrt{T_{max}}$ from all points in the system. This translates, upon conformal mapping, to a certain maximum Γ value. How this is calculated is described below on page 229. The point is that such a calculation must be made. However, this also applies to the other two transformations, as will be seen below.

The next conformal map to be developed was that of Amatore and Fosset [52], here to be called **AF**:

$$\begin{aligned} R &= (1-\theta^2)^{\frac{1}{2}} / \cos\left(\tfrac{\pi}{2}\Gamma\right) \\ Z &= \theta \tan\left(\tfrac{\pi}{2}\Gamma\right) \ . \end{aligned} \tag{12.54}$$

The symbols are (here) the same as for MWA, except that the range of θ is from zero to unity, rather than to $\pi/2$. A convenient result of this transformation is that the concentration profile is very simple at steady state for the potential jump system:

$$C(\theta, \Gamma) = \Gamma \ . \tag{12.55}$$

In other words, the profile has contour lines parallel with the base line ($\Gamma = 0$). This should make simulations using this transformation very efficient. However, the above profile holds at steady state only, and when one compares the efficiency of the four transformations at shorter times, they are all about equally efficient, in the sense that they all take about the same amount of computer time to reach a given target accuracy in the calculated current.

Figure 12.7 shows the equivalences for the AF map. This transformation, in the ranges for θ and Γ shown on the right-hand figure, contains the whole semi-infinite space in (R, Z), so no calculation of a maximum Γ is supposedly needed. As explained below on page 229, however, there can be efficiency reasons for calculating a maximum Γ even here. Also note that, contrary to the MWA map, equal intervals in the conformal space produce θ lines with varying (angular) spacing in (R, Z), being more widely spaced near the disk axis. This is undesirable and might reduce accuracy in the discretisations around that axis. A positive point is that the even spacings in the Γ direction produces expanding distances from the electrode as we move further out. This is desirable, and better than MWA, where the distances expand to a lesser degree. The problem with the angular spacing was overcome by the fourth transformation, to be mentioned below.

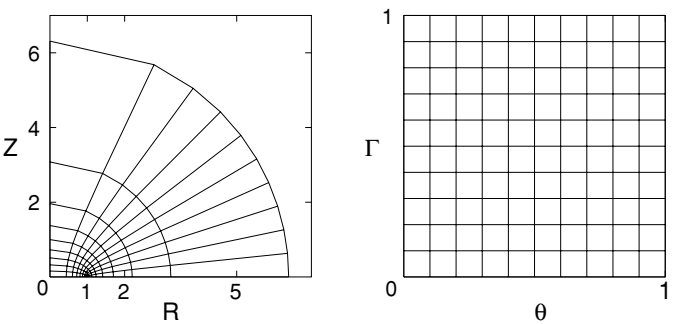


Fig. 12.7. A 10×10 grid in AF (θ, Γ) space and its equivalent in (R, Z)

Next, Verbrugge and Baker [557] changed the definition of Γ in MWA and arrived at the new equation pair, here referred to as **VB**:

$$\begin{aligned} R &= \cos\theta \cosh\left(\frac{\Gamma}{1-\Gamma}\right) \\ Z &= \sin\theta \sinh\left(\frac{\Gamma}{1-\Gamma}\right) \ . \end{aligned} \tag{12.56}$$

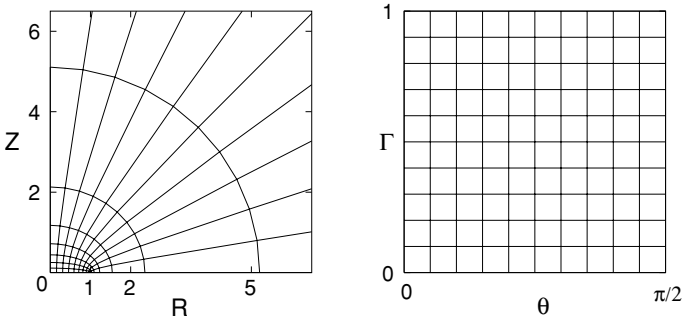


Fig. 12.8. A 10×10 grid in VB (θ, Γ) space and its equivalent in (R, Z)

This transformation produces the equivalence pair in Fig. 12.8. As with AF, the maximum $\Gamma = 1$ completely encloses the semi-infinite diffusion space. The parameter θ has the same limits as with MWA. We note an even spacing of the angles with changing θ and an outwardly expanding spacing for a regular increase in Γ. So this transformation has both positive features.

Recently, a fourth transformation has appeared, that of Oleinick et al. [428], here to be called **OAS**. It is a variant of AF, and is defined as follows:

$$\begin{aligned} R &= \sin\left(\tfrac{\pi}{2}\theta\right) / \cos\left(\tfrac{\pi}{2}\Gamma\right) \\ Z &= \cos\left(\tfrac{\pi}{2}\theta\right) \tan\left(\tfrac{\pi}{2}\Gamma\right) \ . \end{aligned} \tag{12.57}$$

This is the AF map (now only "quasiconformal", as the authors note), with θ replaced by $\cos(\theta)$. Figure 12.9 shows the result. The change from θ to $\cos(\theta)$ eliminates the rather uneven spread of angles seen in Fig. 12.7; the angles are now spread more like those for VB, Fig. 12.8. One difference here is that the angle θ is now zero on the axis, and unity on the insulating plane, the reverse of all three earlier transformations. This transformation appeared to give rather good results, although one expects it to perform somewhat like VB.

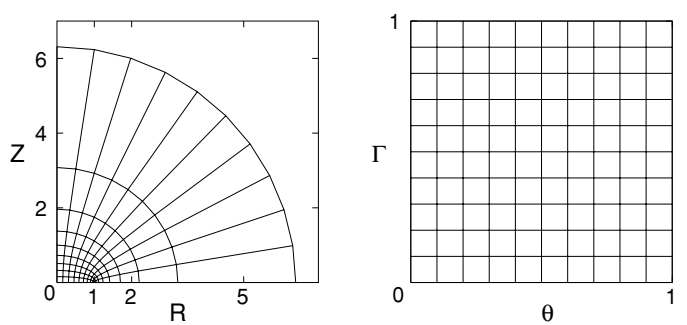


Fig. 12.9. A 10×10 grid in OAS (θ, Γ) space and its equivalent in (R, Z)

Inversion of the Transformations

It is sometimes of interest to invert the transformations, computing the pair of (θ, Γ) coordinates from a given pair (R, Z). All four transformations can be inverted trigonometrically. The inversion for AF has been presented by Svir and Oleinick [540] (with a small typographical problem), and that for OAS by those authors themselves [428]. For some inversions, there are alternative expressions, and there are cases where special formulae must be applied, as is described below.

The inversion equations for the MWA transformation (12.53) are given by

$$\begin{aligned} \Gamma &= \operatorname{arcsinh} \sqrt{\frac{1}{2}\left(R^2 + Z^2 - 1 + \sqrt{(R^2 + Z^2 - 1)^2 + 4Z^2}\right)} \\ \theta &= \arccos\left(\frac{R}{\cosh \Gamma}\right) \end{aligned} \tag{12.58}$$

for both $R > 0$ and $Z > 0$. There is an expression for θ independent of that for Γ, but the one given here is preferable, because the other has two possible solutions, and it is not immediately obvious which one is correct. The one given here comes directly from the first of the transformation pair (12.53). Special cases are

$$\begin{aligned} &R > 0,\ Z = 0: \quad \theta = 0; \quad \Gamma = \operatorname{arccosh} R \\ &R = 0,\ Z > 0: \quad \theta = \pi/2;\ \Gamma = \operatorname{arcsinh} Z \\ &R = 0,\ Z = 0: \quad \theta = \pi/2;\ \Gamma = 0\,. \end{aligned} \tag{12.59}$$

Inversion of VB (12.56) is almost the same as that for MWA, except that the expression for Γ in (12.58) is now an expression for Γ', to be converted to the present Γ by

$$\Gamma = \Gamma'/(1 + \Gamma')\,. \tag{12.60}$$

The AF (12.54) general inversion is [540]

$$\begin{aligned} \theta &= \sqrt{\frac{1}{2}\left(1 - R^2 - Z^2 + \sqrt{(R^2 + Z^2 - 1)^2 + 4Z^2}\right)} \\ \Gamma &= \frac{2}{\pi} \arctan\left(\frac{Z}{\theta}\right) \end{aligned} \tag{12.61}$$

with the special cases

$$\begin{aligned} &R > 1,\ Z = 0: \quad \theta = 0; \quad \Gamma = \frac{2}{\pi} \arccos\left(\frac{1}{R}\right) \\ &R = 0,\ Z > 0: \quad \theta = 1;\ \Gamma = \frac{2}{\pi} \arctan Z \\ &R = 0,\ Z = 0: \quad \theta = 1;\ \Gamma = 0\,. \end{aligned} \tag{12.62}$$

For OAS (12.57) we have

$$\theta = \frac{2}{\pi}\arccos\left(\sqrt{\frac{1}{2}\left(1 - R^2 - Z^2 + \sqrt{(R^2+Z^2-1)^2 + 4Z^2}\right)}\right)$$
$$\Gamma = \frac{2}{\pi}\arctan\left(\frac{Z}{\cos(\frac{\pi}{2}\theta)}\right) . \tag{12.63}$$

Alternative inversion expressions are given by Oleinick et al. [428], equivalent to those above.

The special cases for the OAS inversion are

$$\begin{aligned} R > 1,\ Z = 0: &\quad \theta = 1; &\quad \Gamma = \frac{2}{\pi}\arccos\left(\frac{1}{R}\right) \\ R = 0,\ Z > 0: &\quad \theta = 0; &\quad \Gamma = \frac{2}{\pi}\arctan Z \\ R = 0,\ Z = 0: &\quad \theta = 0; &\quad \Gamma = 0 . \end{aligned} \tag{12.64}$$

The Diffusion Equation in the Mapped Spaces

The transformations lead to a change in the diffusion equation and boundary conditions, in terms of the new variables. The general form of the diffusion equation, for all cases, is

$$\frac{\partial C}{\partial T} = \frac{1}{F}\left(a_\theta \frac{\partial^2 C}{\partial \theta^2} + b_\theta \frac{\partial C}{\partial \theta} + a_\Gamma \frac{\partial^2 C}{\partial \Gamma^2} + b_\Gamma \frac{\partial C}{\partial \Gamma}\right) \tag{12.65}$$

with Tables 12.1 and 12.2 showing the parameters (they are not all constants). MWA and VB (Table 12.1), and AF and OAS (Table 12.2). Note the two terms in the OAS column, that have $(2\Gamma')$ as argument, rather than the usual Γ'.

Consider again Fig. 12.8. It shares with the other two mappings the following simple equivalences in (R, Z) space. The base line of the mapped (right-hand) grid corresponds to the electrode itself, going inward from the edge ($\theta = 0$) to the disk centre (maximum θ). The left-hand edge ($\theta = 0$)

Table 12.1. Parameter values for the diffusion equations in the MWA and VB spaces

Parameter	MWA	VB
F	$\sin^2\theta + \sinh^2\Gamma$	$\sin^2\theta + \sinh^2(\frac{\Gamma}{1-\Gamma})$
a_θ	1	1
b_θ	$-\tan\theta$	$-\tan\theta$
a_Γ	1	$(1-\Gamma)^4$
b_Γ	$\tanh\Gamma$	$(1-\Gamma)^2\tanh(\frac{\Gamma}{1-\Gamma}) - 2(1-\Gamma)^3$

Table 12.2. Parameter values for the diffusion equations in the AF and OAS spaces. For better readability, the symbols $\theta' \equiv \frac{\pi}{2}\theta$ and $\Gamma' \equiv \frac{\pi}{2}\Gamma$ are used

Parameter	AF	OAS
F	$\theta^2 + \tan^2 \Gamma'$	$\frac{\pi^2\left[1-\sin^2\theta'\cos^2\Gamma'\right]^2}{4\cos^2\theta'\cos^2\Gamma'+\sin^2\theta'\sin^2(2\Gamma')}$
a_θ	$1-\theta^2$	1
b_θ	-2θ	$\frac{2\pi\left[1-\sin^2\theta'\cos^2\Gamma'\right]\cos^2\Gamma'\cot\theta'}{4\cos^2\theta'\cos^2\Gamma'+\sin^2\theta'\sin^2(2\Gamma')}$
a_Γ	$\frac{4}{\pi^2}\cos^2\Gamma'$	$\cos^2\Gamma'$
b_Γ	0	0

traces the insulating plane away from the disk edge; while the right-hand edge traces the axis itself. The top line is at maximum distance from the electrode. In the case of AF, VB and OAS, if it lies at $\Gamma = 1$, it corresponds to infinity. In the case of MWA, if suitably chosen, it lies at a distance sufficient for significant diffusional changes to be confined within that limit. More will be said about the limit below.

Equation (12.65) must be accompanied by boundary conditions. These are, generally, now again for the potential jump experiment:

$$\begin{aligned} t \le 0,\ \text{all}\ \theta, \Gamma: &\quad C = 1 \\ t > 0, \Gamma = 0: &\quad C = 0 \\ \Gamma = \Gamma_{max}: &\quad C = 1 \\ \theta = 0, \theta_{max}: &\quad \frac{\partial c}{\partial \theta} = 0\,. \end{aligned} \tag{12.66}$$

The second boundary condition will change appropriately if another experiment than the potential jump is simulated. Here, Γ_{max} and θ_{max} depend on the conformal map used, and on how Γ_{max} is chosen (see below). For MWA and VB, $\theta_{max} = \pi/2$, while for AF, it is unity.

Current Integration in Conformal Coordinates

The current integration (12.19) depends on the transformation. For the three conformal mappings described above, the new expressions are as follows. For MWA [394] and VB [557],

$$I = \frac{\pi}{2}\int_0^{\pi/2} \left.\frac{\partial C}{\partial \Gamma}\right|_{\Gamma=0} \cos\theta\, d\theta\,. \tag{12.67}$$

For AF it is [46]

$$I = \int_0^1 \left.\frac{\partial C}{\partial \Gamma}\right|_{\Gamma=0} d\theta \tag{12.68}$$

and for OAS [428],

$$I = \frac{\pi}{2} \int_0^1 \left. \frac{\partial C}{\partial \Gamma} \right|_{\Gamma=0} \sin\left(\frac{\pi}{2}\theta\right) d\theta \,. \tag{12.69}$$

Choice of Γ_{max}

When simulating the UMDE using one of the transformations, it is often of advantage to know a maximum Γ value, Γ_m. In the case of MWA, this is indeed necessary, as for that transformation, Γ increases indefinitely with increasing distance from the electrode. All the other transformations have a limiting Γ value of unity, corresponding to points at infinity. However, even in these cases, computing time can be saved by restricting the range of Γ or, conversely, if using a fixed number of intervals in the Γ direction, better resolution can be achieved by the restriction. It is thus of interest to find these maximum Γ values.

We proceed from (R, Z) space, where it is easy to define an envelope that encloses the diffusion field to a good approximation. This is the length L, defined, for times T other than very large (see below), by

$$L = 6\sqrt{T} \,. \tag{12.70}$$

For the Cottrell system, this is the distance, beyond which changes greater than 10^{-4} relative to the bulk concentration, are no longer observed. Figure 12.10 shows this line. In the range $0 \le R \le 1$, it is simply the line

$$Z = L \,, \tag{12.71}$$

and for $1 < R \le 1 + L$, it is a quarter-circle defined by

$$Z^2 + (R-1)^2 = L^2 \,. \tag{12.72}$$

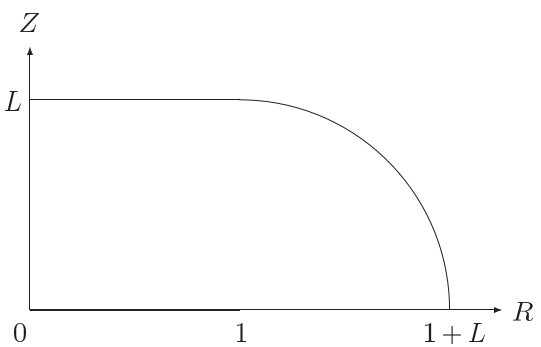


Fig. 12.10. Diffusion limit line

As will be seen, the line generates a corresponding line of Γ values by inversion of a given transformation; and one chooses the maximum value to be on the safe side. That value, for all four transformations, lies at the point $(R, Z) = (1 + L, 0)$, as will now be shown.

To find the maximum Γ value, we first find the maximum value along the straight line segment, defined by (12.71). This equation is substituted in the equation for Z in the given transformation. For example, for the MWA transformation, this leads to

$$\sinh(\Gamma) = \frac{L}{\sin(\theta)} \; . \tag{12.73}$$

This is maximum for a minimum θ, meaning the point (1,L), at which the line joins the quarter-circle. It remains to search that arc for Γ values. Equation (12.72) is substituted by the transformed expressions; for MWA, this produces

$$\sin^2(\theta)\sinh^2(\Gamma) + (\cos(\theta)\cosh(\Gamma) - 1)^2 = L^2 \; . \tag{12.74}$$

Some trigonometric substitutions lead to the result

$$\cosh(\Gamma) = L + \cos(\theta) \tag{12.75}$$

and clearly this is maximum for $\theta = 0$, that is at the point $(1 + L, 0)$. Finally, this yields the maximum value

$$\begin{aligned} \Gamma_m &= \operatorname{arccosh}(1 + L) \\ &= \ln\left(1 + L + \sqrt{(1 + L)^2 - 1}\right) \; . \end{aligned} \tag{12.76}$$

A similar treatment for the other transformations leads, in every case, to quadratic equations in a Γ term, and one of the roots is obviously to be discarded. For both the AF and the OAS transformations, the solution is then

$$\Gamma_m = \frac{2}{\pi} \arccos\left(\frac{1}{1 + L}\right) \tag{12.77}$$

(note that for OAS, θ runs clockwise from the axis, so that at the point $(R, Z) = (1 + L, 0)$, it is equal to $\pi/2$, rather than zero as for the other transformations).

The VB case is obtained from that of the MWA solution, by the substitution

$$\Gamma_{m(\mathrm{VB})} = \frac{\Gamma_{m(\mathrm{MWA})}}{1 + \Gamma_{m(\mathrm{MWA})}} \; . \tag{12.78}$$

The above implies that Γ_m increases indefinitely with T. However, this ignores the fact of a steady state at the UMDE at long times, so for long times, Γ_m might be an overestimate. The choice of L in (12.70) is made on the basis of the Cottrell experiment at a planar shrouded electrode, and

defines the point where the concentration deviates from that in the bulk by no more than 10^{-4}. At the steady state for the UMDE, we have the analytical solution for the concentration profile [184, 490]:

$$C = 1 - \frac{2}{\pi} \arcsin\left(\frac{2}{\sqrt{Z^2 + (1+R)^2} + \sqrt{Z^2 + (1-R)^2}} \right) \tag{12.79}$$

and if we substitute $C = 0.9999$ in this equation, we obtain a curve very close to to a quarter-circle, with a maximum Γ value of $\frac{2\times 10^4}{\pi}$. For all but the open-ended MWA transformations, this means that for roughly $T > 1000$, little is to be gained by not including the whole Γ range. For the MWA transformation, the steady-state Γ_m comes to about 10, so here the rule might be that if a given inversion results in a Γ_m value greater than this, it can safely be reduced to it.

As an aside, in the light of the above, it might be considered that the transformations which enclose an infinite diffusion space are not as advantageous as one might have thought. They appear to solve the problem of needing to estimate Γ_m, but now we see that it is still a good idea to do this. This aspect is however not the most important one. The four transformations perform about equally well when compared for efficiency.

Discretisation

Taking one of the conformally mapped grids, such as the VB grid in Fig. 12.8, it now remains to develop the discretisations of the corresponding *pde* (12.65), with the coefficients as in Tables 12.1 and 12.2, and boundary conditions, as in the set (12.66). One needs to choose the number of points for the derivative approximations. This has been experimented with [532] and the conclusion was that three-point formulae (that is, three-point in each of the two dimensions) gave slightly better results than using a higher number of points at short times, while four or more points were slightly better at longer times. The differences were not great and in that case it might be preferable to use three-point formulae, as they are simpler.

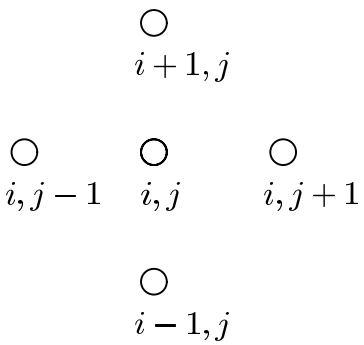
The bottom edge of the right-hand grid in Fig. 12.8 corresponds to the electrode, where values either are simply set (the Cottrellian case) or are computed in some other way (for example, for a reversible reaction). The top edge is defined as points with constant, bulk, concentration. Thus, the points to be treated by discretisation are those in between these two lines. As with direct discretisation on the grid in (R, Z), all points are taken as unknowns and mapped into one long array. Let there be N_θ intervals of length $\delta\theta$ in the θ direction, so that the points are indexed $0 \le j \le N_\theta$, and similarly a number N_Γ intervals of length $\delta\Gamma$, $0 \le i \le N_\Gamma$ along Γ. The total number of points is then

$$N = (N_\theta + 1)(N_\Gamma + 1)\ . \tag{12.80}$$

Then the concentration point at (i, j) maps into a k given by

$$k = i\,(N_\theta + 1) + j + 1\ . \tag{12.81}$$

A stencil of five points is involved in the interior of the grid, conveniently numbered as follows



Each of the positions maps into a k value, the index of the element in the unknowns vector to be solved for. It seems unnecessary to depict these. At the left- and right-hand edges, the zero-gradient in the θ direction is applied, simply as a three-point one-sided approximation, so the edge point expressions result in only three entries in the matrix.

For an internal point at indices (i, j), (12.65), upon discretisation of the right-hand side, becomes

$$\begin{aligned}\frac{\partial C}{\partial T} = \frac{1}{F}\Bigg(& a_\theta \frac{C_{i,j-1} - 2C_{i,j} + C_{i,j+1}}{\delta\theta^2} + b_\theta \frac{-C_{i,j-1} + C_{i,j+1}}{2\delta\theta} \\ & + a_\Gamma \frac{C_{i-1,j} - 2C_{i,j} + C_{i+1,j}}{\delta\Gamma^2} + b_\Gamma \frac{-C_{i,j-1} + C_{i,j+1}}{2\delta\Gamma}\Bigg)\end{aligned} \tag{12.82}$$

from which we can proceed, having decided on a time-integration method. The program `UMDE_VB` (Appendix C) is an example of using the Verbrugge-Baker transformed grid to simulate the potential jump experiment at a disk electrode. BI was used as the first step, followed by 3-point BDF.

12.3.3 A Remark on the Boundary Conditions

In the set of boundary conditions above, (12.3) and (12.66), there are zero-gradient conditions. In the case of the grid in (R, Z), there are two of them, at $R = 0$ and at $(Z = 0, R > 1)$. Although both are given by Crank and Furzeland [184], these authors do not in fact use them both *as* boundary conditions; the one at $R = 0$ is used as a symmetry argument, in order

to develop the form of the term that might become singular, arriving at (12.42). Thus, they allow diffusion along and away from the axis. Similarly, one might allow diffusion in the radial direction along the insulating plane, as well as normal to it, leaving out the (zero) normal $\partial C/\partial Z$ term, and using symmetry to construct the special form of $\partial^2 C/\partial Z^2$ there. This latter discretisation seems not to be used by anyone. What is used is simply the no-flux condition, discretised suitably. One might suspect that this makes use of less information than is available, and thus renders the solution less accurate. However, see below.

In the case of the mapped grid, we also have two zero gradients $\partial C/\partial\theta$, at both the left- and right-hand edges of, for example, Fig. 12.8. This is the way it is usually done [52,532,557]. At both edges, however, it is also possible – and might make more sense – to invoke the diffusion equation, taking symmetry into account, and leaving out the first derivative terms $\partial C/\partial\theta$ there. The program `UMDE_VB` was modified with this in mind, allowing diffusion in both directions. Some numerical experiments showed that the results were almost the same as for using the boundary condition, and convergence to an accurate value with increasing grid intervals was no faster. Therefore the choice remains a personal one.

13 Convection

Convection has long been coupled with electrochemistry, and the name hydrodynamic voltammetry has become standard. In electroanalytical chemistry we mainly seek reproducible conditions. These are almost always attained by systems in which a steady convective state is achieved, although not always. Thus, the once popular dropping mercury electrode (see texts such as [74, 257]) has convection around it, but is never in steady state; it might be called a reproducible periodic dynamic state.

The focus in this chapter is on channel electrodes, which are popular at this time, and the way to simulate them illustrates the method generally.

13.1 Some Fluid Dynamics

Fluid flow, since it transports material, is enmeshed with diffusion in electrochemical cells. Some basics are therefore in order here.

Useful fluid dynamic systems are partially enclosed systems. Consider the open system consisting of an infinite solid plate at the bottom of a semi-infinite fluid, all at rest. Now, at $t = 0$, let the plate start moving with a certain velocity, in the direction of its plane. If one follows the fluid velocity this generates as a function of time and distance from the moving plate, it is seen that the equation governing this process is of the same form as the diffusion equation, and the solution is mathematically the same as for the Cottrell system, as is shown in the first few pages of standard texts dealing with fluid flow [120, 498]. This, then, is not a system of great use to electrochemists, since it does not lead to a steady state flow distribution. Consider now Fig. 13.1. It shows a sideways view of a channel or slit, of height $2h$, and a depth (into the paper) so great that there are assumed to be no gradients in that direction. The y-axis has its origin in the central plane, indicated by the dashed line. If we ignore entry effects at the inlet end of this channel (but see below) and if the flow is laminar (that is, not turbulent), then there is a steady state flow, with no velocity components in the y-direction. The component v_x in the x-direction will then be a known function of y. At the walls ($y = \pm h$), the fluid clings to the solid surface, that is $v_x(\pm h) = 0$. As can be shown [120, 498], the velocity profile is of the parabolic form

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 235–246 (2005)
www.springerlink.com

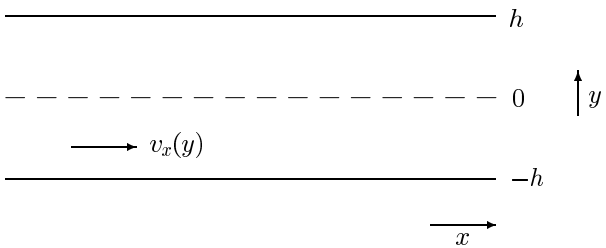


Fig. 13.1. Flow through a channel

$$v_x(y) = v_0 \left(1 - \left(\frac{y}{h}\right)^2\right) \tag{13.1}$$

where v_0 is the velocity along the central plane at $y = 0$. It is the maximum velocity in the channel, and is given by

$$v_0 = \frac{h^2}{2\mu} \frac{dp}{dx} , \tag{13.2}$$

in which μ is the fluid's viscosity and dp/dx is the pressure gradient driving the flow. Another quantity of interest is the mean flow velocity v_m through the slit,

$$v_m = \tfrac{2}{3}\, v_0 . \tag{13.3}$$

The above holds only for laminar flows, which means that the Reynolds number is sufficiently small. It is defined as

$$\mathrm{Re} = \frac{v_m L}{\nu} \tag{13.4}$$

where ν is the kinematic viscosity, equal to μ/ρ, ρ being the fluid density. L is a characteristic length pertaining to the flow, here equal to $2h$, the channel height. For the flow to be laminar, the Reynolds number should be smaller than a few 1000. This is to some extent uncertain, depending on the smoothness of the walls and the way in which the fluid enters the slit.

For electrochemical purposes, where electrodes are (usually) embedded in the channel bottom, it is convenient to shift the y coordinate so that y is zero at the channel bottom. The equation for the velocity profile then changes to

$$v_x(y) = v_0 \left(1 - \left(\frac{y - h}{h}\right)^2\right) \tag{13.5}$$

(v_0 of course remaining the same, as in (13.2)).

Since the velocity of flow has a parabolic function, the velocity profile near the walls is nonlinear. However, in many works, this is approximated by a linearised form, as the gradient right at the walls. This makes the mathematical analysis of diffusion near one of the walls easier. Differentiating (13.1) and setting $y = -h$ (that is, considering the bottom surface), we obtain

$$\left.\frac{dv_x}{dy}\right|_{y=-h} = \frac{2v_0}{h} = \frac{h}{\mu}\frac{dp}{dx} \tag{13.6}$$

so that near the channel bottom ($y = 0$, having now moved the y-axis) the velocity profile is

$$v_x(y) \approx \frac{2v_0}{h}\, y\ . \tag{13.7}$$

This system is currently one of the most used hydrodynamic cell types, with electrodes embedded in the surfaces.

Another geometry, not quite as popular for practical reasons but perhaps equally useful, is a tube, shown in Fig. 13.2. Now the centre is the axial line $r = 0$ and the walls are at $r = R$. The laminar flow for this is given by

$$v_x(r) = v_0\left(1 - \left(\frac{r}{R}\right)^2\right) \tag{13.8}$$

where v_0 is the velocity along the axis, $r = 0$. It is

$$v_0 = \frac{R^2}{4\mu}\frac{dp}{dx} \tag{13.9}$$

and the average flow velocity through the tube is

$$v_m = \tfrac{1}{2}\, v_0\ . \tag{13.10}$$

The Reynolds number Re is here defined by setting L in (13.4) equal to the tube diameter, $2R$. As for the channel, Re must not exceed a magnitude of 1000 for the flow to be laminar. The velocity gradient at the wall is very similar to that for the slit (13.6), with y replaced by r and h by R.

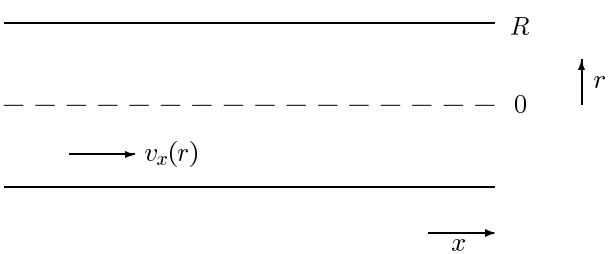


Fig. 13.2. Flow through a cylindrical tube

When using a tube or channel to establish a laminar flow, one must be aware of entry effects. At the entry point of the tube or channel, the flow velocity profile will be even across the cross section of the tube. As the flow moves into the tube, the profile gradually becomes parabolic. So, if we rely on its being parabolic, the electrodes must be placed sufficiently far downstream

for that to be true. Kay and Nedderman [333] and Schlichting [498] both provide (almost) the same formula for the tube, expressed here as

$$L = 0.06\, R\, \mathrm{Re}\ , \tag{13.11}$$

so that electrodes must be at least a distance L into the tube. The situation in a channel is similar, with h replacing R, as has been shown [499]. The value of the constant depends on the extent to which the profile is to be established. At 0.06, this is about 95% in terms of the flow velocity at the centre of the tube or channel. Prandtl and Tietjens [448] state a much larger constant, 0.6, based on a 99% convergence to fully parabolic flow.

Another flow system that has had some use in the past is a jet impinging on a flat wall, shown in Fig. 13.3. There is a narrow jet of fluid flowing downwards out of an orifice in the top wall, hitting the bottom plate. The figure shows some flow lines, which go both in the vertical and horizontal directions. At the point P there is no flow; this is the stagnant point. Electrodes can be placed on or around this point. The flow distribution has been mathematically solved by Glauert [269], whose solutions were extended by Albery and Brett [30], with an empirical constant being provided by Yamada and Matsuda [579]. The profiles are quite complex and will not be gone into here.

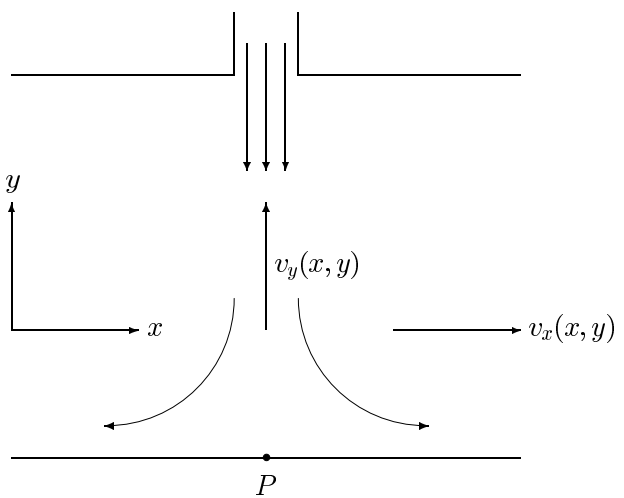


Fig. 13.3. Flow in the wall jet system

Other convective systems have been used in electroanalytical chemistry. The oldest one is the dropping mercury electrode [74, 257]. Convection here arises by virtue of the expansion of the growing mercury drop, and the transport equation is pleasantly simple and unidimensional for the simplified case,

assuming a spherical drop that is not falling downwards, and assuming that the sphere is large compared with the diffusion later thickness. This electrode is no longer in wide use and will not be further pursued here.

Another system is the family of rotating electrodes. These are disks and/or rings mounted concentrically with the axis at the end of a cylindrical rod, rotating in an electrolyte. Making some reasonable simplifications, this system also gives rise to a unidimensional transport ([362]), and the velocity distributions around the end of the rod were first derived by von Kármán [564] and Cochran [166], later to be improved by Sparrow and Gregg [516]. Levich solved the case of steady state limiting current (see [362]), using some approximations, later corrected by Gregory and Riddiford [281]. Details can be seen in Bard and Faulkner [74].

13.1.1 Layer Relations

In Chap. 2, the concept of the diffusion layer was established. It is a thickness, within which a large fraction of diffusional changes take place, and at a distance of several times this thickness, practically no more diffusional changes are observed. This layer will here be given the symbol δ_D (D for diffusion). In fluid dynamics, there is a similar layer, within which most of the velocity changes occur. This is the hydrodynamic layer δ_h. It turns out that for diffusive mass transfer, δ_D is usually much smaller than δ_h. This is fortunate, because it justifies to some extent the linearised velocity profiles often assumed near walls, making analysis easier. These relations are very lucidly discussed in a classic paper by Vielstich [560].

13.2 Electrodes in Flow Systems

Electrodes have been placed in many flow system geometries. The rotating disk or ring-disk electrodes are well known. Narrow rings have been mounted flush inside tubes [86,247,325,578], on rotating electrodes such as the classical rotating disk electrode (RDE) mentioned above, treated by Levich [362] and the rotating ring-disk electrode (RRDE) extensively described by the Albery group [29, 31, 32, 33, 34] and other variants and applications, too numerous to mention here. Reviews such as that of Albery et al. [37], Penar [439] and Williams and MacPherson [573] (on modulated flows) discuss these, providing many references. Electrodes, both singly [30, 55, 579], as ring-disk [256] or (for the wall jet case) even groups of separate electrodes [555] have been mounted as targets of the flow.

Single electrodes in a flow where a steady state is attained act much like the DME or RDE, in that a sigmoid current/voltage curve is measured, from which information about the electrochemical reaction can possibly be gleaned. Heterogeneous rate constants can for example be measured if the flow is sufficiently fast. This was the aim of Bernstein et al. [86] and their

turbulent flow in a tube with a ring electrode, and equally intense turbulence is generated at an electrode positioned close to an ultrasonic horn [69]. There are too many references of this kind to mention here. Another objective can also be a reverse of the usual electrochemical aim: so-called electrochemical probes have been used to measure flow rates [80, 399].

Double electrodes were suggested, as an added ring outside the central disk on an RDE, by Frumkin et al. [256] and as a second embedded strip downstream of the first in a channel, suggested first by Gerischer et al. [267]. Here, the idea is to produce a substance by electrolysis at the upstream electrode, and to detect it at the other electrode downstream. One speaks of the collection factor N, the ratio of the detector current to the generator current. The symbol was first used by Frumkin et al. Much theory has been presented for the many possible geometries. A very general theory was worked out by Matsuda [388], and also, for what they called the "dimicroelectrode", by Kermiche et al. [335]. Gerischer et al. also attempted a rough first treatment [267], using (as did Kermiche et al.) a linearised velocity profile near the electrode for simplicity. Solutions are not easy to obtain in this area, so this is an intense application of digital simulation.

Another application of double, generator/collector, electrodes is what is called diffusion layer titration. This can be used for a quantitative analysis of some species in solution. The technique was first suggested by Bruckenstein and Johnson [157], and has been followed up since then, with theory [33, 458] and simulations [81, 458, 541, 554] (naming just a selection of works).

It may be added that generator/collector cells can be implemented without convection. Double microbands [65] and interdigitated bands [294, 447, 500] have been considered for this purpose, for titrations [458, 541] and for studies in electrochemically generated chemiluminescence (ECL) [539]. See also the review by Amatore [46] with more references therein.

13.3 Simulations

The earliest simulations of convective systems were those of the DME [146, 229, 230, 487] and the RDE and RRDE. Prater and Bard performed the first simulations of the RRDE [449,450,451], using the explicit box method. Maloy et al. simulated ECL at an RRDE. Margarit et al. simulated a ring-ring electrode [380,379] and studied collection factors by simulation; Clarenbach et al. [164,165] simulated their own modification of the RRDE, also simulating fluid flow around it, as did Mandin et al. [378], using a program package. Feldberg [235] used hopscotch on a RRDE, Nolan [420] used OC. Balslev and Britz [68] used a brute force method to compute the steady state at an RDE with a complex reduction mechanism. Dan et al. [192] applied CN and what they called MDUM, a multigrid method, simulating transients at an RDE, which had been done earlier by Strutwolf [529, 533, 534]. This list is by no means exhaustive; only some representative examples have been cited here.

In flow systems that necessitate consideration of two-dimensional geometry, Flanagan and Marcoux did some early work [247]. They examined a variety of conditions, among them the importance of axial diffusion in a tube. They found that neglecting axial diffusion is justified for most flows except the slowest. This is because transport due to the flow dominates in the axial direction, and this holds for electrode lengths that are small compared with the tube radius. This is often called the Levich approximation. Levich [362] related the diffusion layer thickness to the tube radius. It is a function of distance x along the electrode and flow velocity,. The condition can then be reduced to the condition

$$\frac{xD}{v_0 R^2} \ll 1 \tag{13.12}$$

which limits the length of the electrode along the length coordinate x. This is referred to in Wu [578] and the same condition was given by the early simulation papers of Albery and coworkers [35,36], who however do not cite Levich. Albery et al. [35] are interesting in that they present an early finite difference simulation in this context, and use some coordinate stretching by transformation as well.

Among the many papers written on the simulation of band electrodes embedded in a channel flow system, the one by Anderson and Marcoux [54] was the first. They simulated a single band in a channel, and experimented with the explicit, trapezoidal (CN) and BI techniques. They concluded that BI is probably the best. This has since been the most used technique in the papers to follow. There are good reasons for this, as outlined clearly by Fletcher [250] and Strikwerda [528]. Fletcher shows that discretisation of convective transport yields stable forms only if what fluid dynamicists call upwinding is used. This amounts to backwards implicit, in the x direction (along the flow), rather than, as in previous chapters, in the time direction. Other algorithms may however be better, as shown by Alhunaizi [44], who considers a certain high-order explicit method the best.

Most of the flow systems used in these channel experiments attain steady state, and, as will be seen below, the x direction can take the place of time. There is a multitude of works on the simulation of the channel electrode system, dominated to a large extent by the Compton school [171,172,174], using BI. Many other references can be found by a library search on the name of Compton, up to the present.

13.4 A Simple Example: The Band Electrode in a Channel Flow

There is clearly a multitude of hydrodynamic systems of interest, and the focus here is on the channel flow system with a single narrow band embedded in the channel floor. The methods of discretising this system point the way

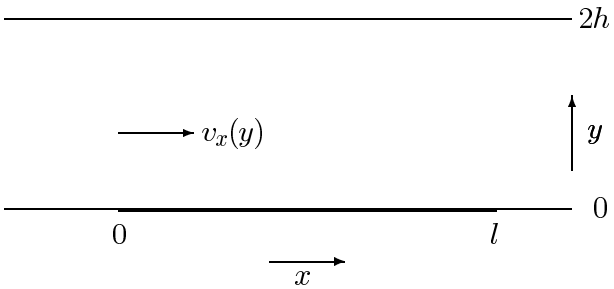


Fig. 13.4. Band electrode in a rectangular channel

to other systems. Consider Fig. 13.4. We want the current over a short band in a low wide channel, as shown in the figure. The channel is of height $2h$ and the band has width l in the x-direction, the direction of flow. The band's length (into the paper) is a, and it is assumed that $a \gg l$. The flow is laminar with velocity $v_x(y)$, a function of y and is given by (13.5). The mean flow velocity is given by (13.3). If the flow is sufficiently fast (as is assumed), then we can ignore diffusion in the direction of the flow, as the flow will dominate transport. The transport equation is then

$$\frac{\partial c}{\partial t} = D\,\frac{\partial^2 c}{\partial y^2} - v_x\,\frac{\partial c}{\partial x}\ . \tag{13.13}$$

13.5 Normalisations

The following normalisations are used. The characteristic time τ is chosen as the time it takes the mean flow v_m to traverse the length of the electrode, l. Thus,

$$\tau = \frac{l}{v_m} \tag{13.14}$$

and so time becomes the normalised T,

$$T = t/\tau = \frac{v_m}{l}\,t\ . \tag{13.15}$$

Distances are normalised by l, so

$$X = x/l\ , \tag{13.16}$$

$$Y = y/l\ , \tag{13.17}$$

$$H = h/l\ . \tag{13.18}$$

Concentrations are referred to the initial bulk value c_b,

$$C = c/c_b \, . \tag{13.19}$$

This leads to the new transport equation,

$$\frac{\partial C}{\partial T} = \frac{1}{\mathrm{Pe}} \frac{\partial^2 C}{\partial Y^2} - V_X \frac{\partial C}{\partial X} \tag{13.20}$$

with

$$V_X = \frac{3}{2} \left\{ 1 - \left(\frac{Y - H}{H} \right)^2 \right\} \tag{13.21}$$

and

$$\mathrm{Pe} = \frac{v_m \, l}{D} \, . \tag{13.22}$$

Pe is a Péclet number, analogous to the Reynolds number, which is defined for this flow as

$$\mathrm{Re} = \frac{2 v_m h}{\nu} \, . \tag{13.23}$$

While Re is the measure of the relative magnitudes of the inertial forces to the viscous forces in the flow, the Péclet number is the measure of the relative magnitudes of transport by convection and diffusion [333]. The length scales used for the two numbers are different (the band height $2h$ for Re, l for Pe).

We need to know how far away, normal to the electrode, we need to compute concentration changes, that is, what value of $Y = Y_m$ is sufficient. This can be estimated in the following manner. The mean flow goes past the electrode in time τ, and in that time, a diffusion layer of height about equal to $\sqrt{D\tau}$ can be attained at the downstream end of the electrode. Taking, as usual, six times this length, we get a maximum y_m of $6\sqrt{D\tau}$. Normalising this by l so that we make $Y_M = y_m/l$ and substituting for τ from (13.14), we have

$$Y_m = 6\sqrt{\frac{D}{l v_m}} = 6/\sqrt{\mathrm{Pe}} \, . \tag{13.24}$$

We now have two situations, with two different boundary conditions. If $2H < 6/\sqrt{\mathrm{Pe}}$, then Y_m must be set equal to $2H$, and we apply a no-flux condition to the channel roof. If however $H > 6/\sqrt{\mathrm{Pe}}$, then we can apply the constant concentration (bulk value) to the level Y_m.

Figure 13.5 shows a rather coarse grid drawn on the system, for the case $2H > 6/\sqrt{\mathrm{Pe}}$. If one wishes only to compute the current, then points downstream from the electrode need not be computed. If concentrations downstream are of interest, the grid must be extended in that direction. The range in direction X is divided into N_X intervals spaced apart by δX, starting from the left-hand boundary one interval upstream of the leading electrode edge. The vertical direction is divided into $N_Y + 1$ horizontal lines spaced δY apart. Giving indices j and i, respectively, to the X and Y direction, we have

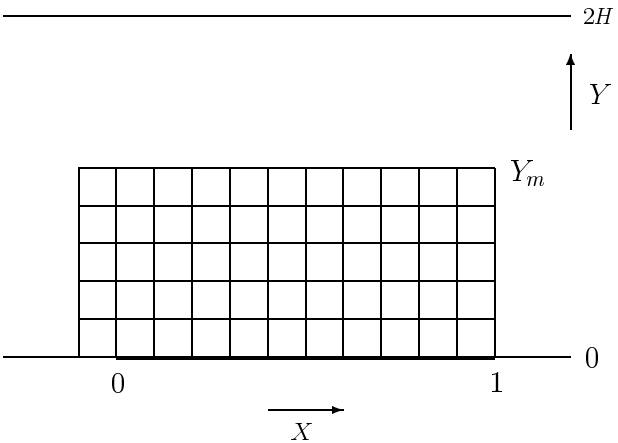


Fig. 13.5. Band electrode system with grid superimposed

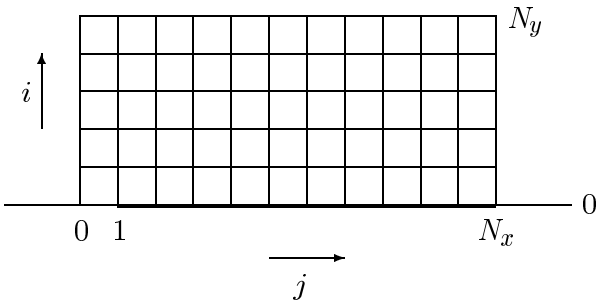


Fig. 13.6. Band electrode system with indexed grid

a working grid as shown in Fig. 13.6. Note that there are two independent dimensionless variables that must be stated for a given simulation. They are the Péclet number Pe, which expresses the magnitude of the flow velocity with respect to diffusion, and H, expressing the half-height of the channel in electrode length units l.

We have two situations. The simpler one is that we only want the steady state current, in which case the time derivative drops out of (13.20), leaving only

$$V_X \frac{\partial C}{\partial X} = \frac{1}{\mathrm{Pe}} \frac{\partial^2 C}{\partial Y^2} \,. \tag{13.25}$$

Boundary conditions are then

$$X < 0 : \; C = 1 \tag{13.26}$$

$$0 \le X \le 1, \; Y = 0 : \; C = 0 \tag{13.26}$$

$$Y = Y_m : \; \begin{cases} C = 1 & (2H \ge 6/\sqrt{\mathrm{Pe}}) \\ \frac{\partial C}{\partial Y} = 0 & (2H < 6/\sqrt{\mathrm{Pe}}) \end{cases} \,. \tag{13.27}$$

Note that (13.25) is parabolic, there being a term on the left-hand side in $\partial C/\partial X$. This suggests a solution analogous to a time march, that is, an X-march, starting from "initial conditions" at the grid line just upstream of the electrode, at $j = 0$ in Fig. 13.6, and moving to the right from there, successively computing each vertical row of points. This was done by Anderson and Moldoveanu [54]. As mentioned earlier, it can be shown that the left-hand term in (13.25) is best discretised as a backward difference (upwinding). It is tempting to apply a central difference form here but this can cause oscillations [250, 528].

Equation (13.25) is now discretised in the following manner,

$$V_X \frac{C_{i,j} - C_{i,j-1}}{\delta X} = \frac{1}{\mathrm{Pe}} \frac{C_{i-1,j} - 2C_{i,j} + C_{i+1,j}}{\delta Y^2} \,. \tag{13.28}$$

Writing

$$\lambda_i = \frac{\delta X}{V_X \delta Y^2 \mathrm{Pe}} \tag{13.29}$$

(recall that V_X is a function of Y and thus varies with index i), we have the discrete form

$$C_{i,j} - C_{i,j-1} = \lambda_i \left(C_{i-1,j} - 2C_{i,j} + C_{i+1,j} \right) \tag{13.30}$$

which is of the form seen in the time-march procedure in Chap. 8 (albeit simpler because here, equal intervals in Y were used), and the solution, by the Thomas algorithm, described in Sect. 8.3, can be used.

The dimensionless current is then given by the integral over the length of the electrode,

$$I = \int_0^1 \left. \frac{\partial C}{\partial Y} \right|_{Y=0} \tag{13.31}$$

which can be implemented as a trapezium or Simpson's rule.

The example program `CHANNEL_BAND` (Appendix C) is a simple implementation of the above, using two-point upwinding and equal intervals. The system has a known solution, given by Levich [173, 267, 362, 387] which implies that the current should be proportional to $\mathrm{Pe}^{\frac{1}{3}}$. This is tested in the program, and found, for a number of runs, to be true for some ranges of parameters.

If time dependence is desired, then the full transport (13.20) must be discretised, and a time-march performed. The problem is then a 2D one. As in the previous chapter on 2D systems, one would spread the grid points into one long vector of unknowns, either by stacking the horizontal grid lines end on end, starting (as for the UMDE system in the previous chapter) with the bottom row, or perhaps the verticals. Given the above description and those in Chap. 12, the development of this is straight-forward and will not be further pursued here.

A remark on the linearisation of the velocity profile V_X is now in order. It clearly applies only for $Y_M \ll H$. Given that one would in any case pre-compute the N_y V_X values, there seems little point in linearisation as in (13.7). This is of greater interest in mathematical analyses but not for simulations.

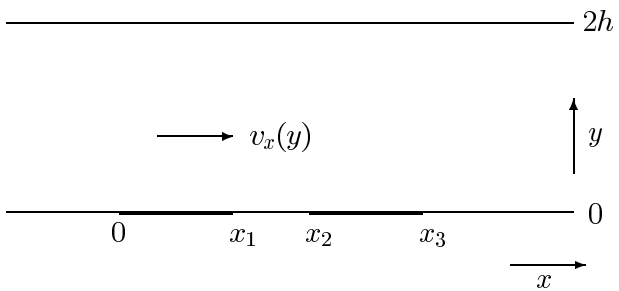


Fig. 13.7. Channel flow with two bands

The same procedure, both for the steady state and the time dependent system, can be extended to the channel with two bands, in generator-collector mode, as shown in Fig. 13.7. There are more boundary conditions, but they are straight-forward to apply. For details, the reader is referred to a series of articles by the Compton group [40, 171, 174, 175, 244, 463] (citing just a selection of a large opus).

14 Performance

In this chapter, the performance of the various methods described is examined. This involves convergence and economy of computer time. Some of the more sensible simulation methods are compared.

14.1 Convergence

The aim of a simulation is to approximate the underlying exact solution as accurately as possible, in a minimum of computer time. Solution is achieved by some discrete formula, which has truncation errors, due to neglect of some (higher) Taylor terms in the discretisation formulae. These errors must become smaller as we make the intervals both in time and space smaller and the errors must, at least, not grow in the course of a number of steps. This property is called convergence. In the limit, as δT and δX (that is, H) approach zero, the errors must also do so. In order for this to happen, two conditions must hold. The first is that the discretisation expression used must be consistent with the differential equation it approximates. The second is that the expression must be stable. This means that an error in the solution at a given step is not amplified by subsequent steps. These two issues will be examined separately.

Generally, it can be said that consistency is not as great a problem as stability, as pointed out by Lapidus and Pinder [350]. Inconsistent discretisations have been devised, but they are rather rare.

Another way of viewing convergence is not whether a given solution converges towards the exact solution, but how it does so. Does it approach smoothly, the errors keeping on one side of the zero line, or does it approach with oscillations? Some (but not all [282]) regard oscillations as a bad thing. This is not necessarily so, as is known from electrical engineering. An optimal control circuit is often the one that responds to a step change in an input with a strongly damped oscillation. A smooth response is slower. However, if the oscillations persist for a long time, they are again not optimal. This is mirrored in simulation in the ways some stable methods behave. BI has what some regard as a pleasant, smooth, approach to the solution, while CN, when using large λ values, oscillates. CN is however the method with higher order of accuracy, and if the oscillations are damped quickly, it is the better

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 247–272 (2005)
www.springerlink.com

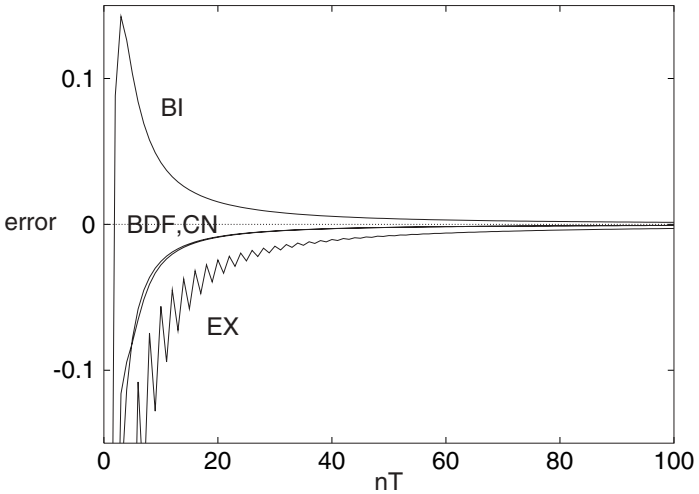


Fig. 14.1. Errors in the simulated Cottrell current, using the indicated methods, with $\lambda = 0.5$

method. It is possible to prevent the CN oscillations by damping them in the first few steps [149, 432], see Chap. 8.

To illustrate the above, consider Fig. 14.1. Four methods were used to compute currents for the Cottrell experiment, all using 100 steps in time up to $T = 1$ and a λ value of 0.5. We note that the explicit method shows some error oscillations (it is at the edge of stability at this λ value). BI approaches a zero error similarly slowly to EX but in a smooth manner. CN and BDF (3-point, using the simple start and the $\delta T/2$ correction) approach zero error about equally fast, and more rapidly than the other two methods. Figure 14.2 shows the same results for $\lambda = 3$. Method EX is now left out, as it is unstable for $\lambda > 0.5$. BI and BDF still show a smooth approach to a zero error (for BDF, after a very few initial oscillations), and CN now starts with some oscillations. These are however quickly damped, and after about 20 steps, CN shows about the same error as 3-point BDF. A further increase in λ, to 10, is shown in Fig. 14.3. The scale needed to be increased for that figure and even so, the initial CN oscillations are so large that they could not be plotted on this scale. It looks as if CN does converge eventually, but a close look at the final results after 100 steps shows that it is now slightly less accurate than BDF, implying that the oscillations even then have not quite been damped out.

This behaviour is typical of CN and known since its inception [185] and is due to the so-called A-stability of the method (see below). As mentioned above, however, there are various means of damping out the oscillations, discussed in Sect. 8.5.1, Chap. 8. The attraction of CN is, of course, that it presents no difficulties with start-up, as does BDF.

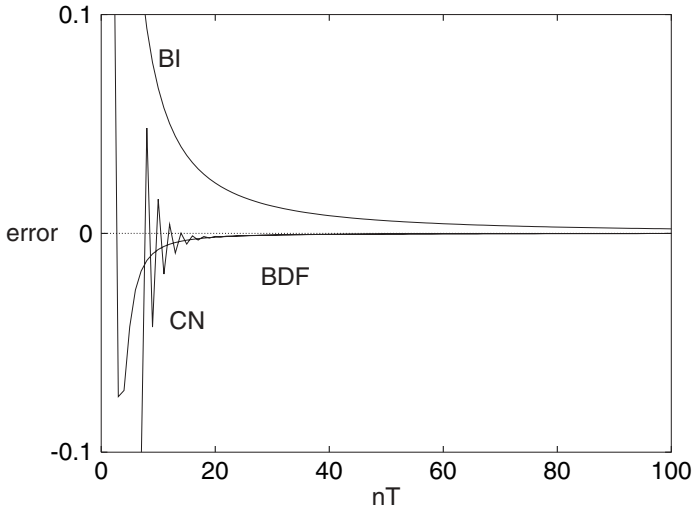


Fig. 14.2. Errors in the simulated Cottrell current, using the indicated methods, with $\lambda = 3$

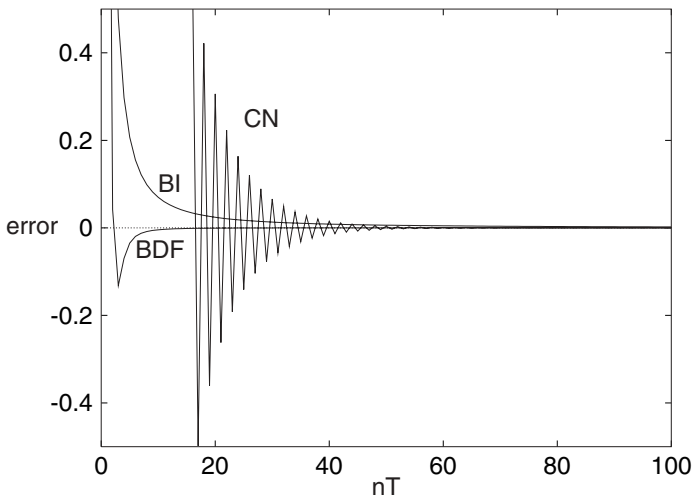


Fig. 14.3. Errors in the simulated Cottrell current, using the indicated methods, with $\lambda = 10$

In what follows, the issues of consistency, stability and efficiency are addressed.

14.2 Consistency

In the present context we have two intervals: δT in time, and H in space. A given discretisation is said to be consistent if, as both of these intervals approach zero, the discretisation approaches the *pde* it is meant to approximate. Take the simple explicit discretisation on equal intervals, in (5.2), which we rearrange into the form

$$C_i' = \lambda C_{i-1} + (1-2\lambda)C_i + \lambda C_{i+1} \; . \tag{14.1}$$

The three terms at a small interval away from C_i can be Taylor-expanded around the value C_i. We have (dropping the index i for the derivatives)

$$\begin{aligned} C_i' &= C_i + \delta T \frac{\partial C}{\partial T} + \frac{\delta T^2}{2}\frac{\partial C^2}{\partial T^2} + O(\delta T^3) \\ C_{i-1} &= C_i - H\frac{\partial C}{\partial X} + \frac{H^2}{2}\frac{\partial C^2}{\partial X^2} - \frac{H^3}{6}\frac{\partial C^3}{\partial X^3} + \frac{H^4}{24}\frac{\partial C^4}{\partial X^4} - O(H^5) \\ C_{i+1} &= C_i + H\frac{\partial C}{\partial X} + \frac{H^2}{2}\frac{\partial C^2}{\partial X^2} + \frac{H^3}{6}\frac{\partial C^3}{\partial X^3} + \frac{H^4}{24}\frac{\partial C^4}{\partial X^4} + O(H^5) \end{aligned} \tag{14.2}$$

where the $O(..)$ terms indicate where the Taylor series were cut off. Inserting these in (14.1) and tidying up, this becomes

$$\frac{\partial C}{\partial T} + \frac{\delta T}{2}\frac{\partial^2 C}{\partial T^2} + O(\delta T^2) = \frac{\partial^2 C}{\partial X^2} + \frac{H^2}{12}\frac{\partial^4 C}{\partial X^4} + O(H^4) \tag{14.3}$$

and it clear that as both δT and H approach zero, the equation approaches the *pde* we are in fact trying to approximate.

Most discrete approximations that have been mentioned in this book are consistent, with the exception of one. This is the DuFort-Frankel method [216], described on page 153 in Chap. 9. It is stable for all λ, yet it has a consistency problem. Giving (9.19) the same treatment as above, one ends with

$$\frac{\partial C}{\partial T} + \frac{\delta T^2}{6}\frac{\partial^3 C}{\partial T^3} + O(\delta T^3) = \frac{\partial^2 C}{\partial X^2} + \frac{H^2}{12}\frac{\partial^4 C}{\partial X^4} - \frac{\delta T^2}{H^2}\frac{\partial^2 C}{\partial C^2} + O(H^4) \; . \tag{14.4}$$

It is seen that again, two of the three remaining Taylor terms vanish as the intervals approach zero, but the one containing $\frac{\delta T^2}{H^2}$ does not. As pointed out by Shih [506] and Strutwolf [529], what is happening here is that the approximation is consistent not with the parabolic *pde*, but with a hyperbolic one instead. For large λ (leading also to large $\frac{\delta T^2}{H^2}$), the DuFort-Frankel method is not suitable for the simulation of parabolic problems. The method has been suggested [233] for use in electrochemical simulations but since Rudolph [476] pointed out its problems, it has not been used again in electrochemistry.

14.3 Stability

The stability of a given simulation method can be defined mathematically rigorously, or more loosely. A loose description might be that given in Smith [514, p. 47], namely that the amplification of initial conditions be limited. This means that if, due to truncation or roundoff, there be errors in, say, the concentration values at a given step, these errors are not amplified without bound in subsequent steps. There are various categories of stability, to be seen in the relevant texts [286, 350, 514] (to cite only three of a multitude of such texts). Below, a rather brief and less technical treatment will be given than is provided in these texts. Several methods for determining stability are described, and some special conditions that can affect stability.

From the range of methods for determining stability of a given algorithm such as EX, CN, BI or BDF, etc., this chapter restricts itself to the heuristic, the Neumann and the matrix methods, as well as a fourth that makes use of the stability function.

Stability can be classified into a number classes. We refer to *conditional* and *unconditional* stability. EX is a conditionally stable method, because there is a restriction on the value of λ, whereas CN and Laasonen are unconditionally stable. There is, however, a difference between their stabilities. Dahlquist [189, 190] and Henrici [302] refer to *weak* or *strong* stability of some methods. This has since been tightened to a large number of subcases of stability, with a rough division between those methods that show **A-stability** and those that show **L-stability**. For more details, see such texts as Smith [514] and Hairer & Wanner [285]. Here it will suffice to describe them in less detail. Methods that are A-stable (such as CN) have some error propagators close to unity in magnitude even for large λ (for error propagation, see the Neumann method, below). L-stable methods, on the other hand, have error propagators that all approach zero as λ grows larger. Laasonen is one such method. We thus prefer our methods to be L-stable.

14.3.1 Heuristic Method

Lapidus and Pinder [350] describe the simplest of all stability determinations and call it the heuristic method. With this method, one tries to compute a few steps in time from a perturbation in initial conditions, and sees how the perturbation is propagated after a number of steps. In Fig 14.4, the explicit method is used, (14.1), setting $\lambda = 0.5$. At some time level $N\delta T$, an error is placed at a point along X in Fig. 14.4. Points marked in the figure without numbers are assumed to have zero values. It is seen that in subsequent steps after the Nth, the error is spread along X, but with decreasing magnitude. It is known that for $\lambda \leq 0.5$, this method is stable, as the figure also suggests. Now consider Fig. 14.5, where the same calculation has been done using $\lambda = 1$. Obviously, the error is quickly amplified, and the method is unstable.

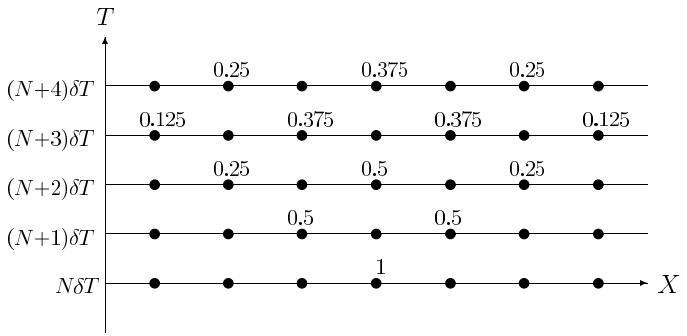


Fig. 14.4. Propagation of a single error for EX with $\lambda = 0.5$

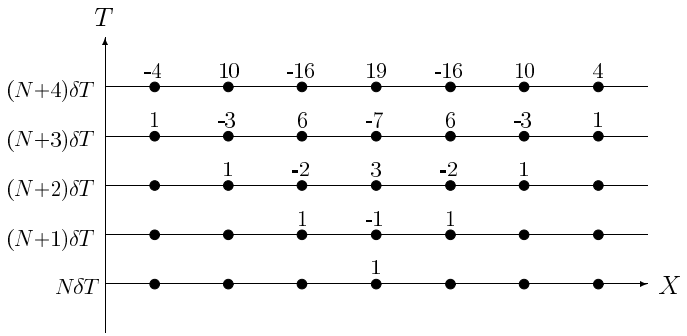


Fig. 14.5. Propagation of a single error for EX with $\lambda = 1$

Planting a perturbation in a simulation can be a useful way of testing for stability, especially under special conditions.

14.3.2 Von Neumann Stability Analysis

A more analytical method of stability analysis is the method of von Neumann [424, 565] (note that [424] is mostly incorrectly cited as being of the year 1951 [139]). The method focusses on an interior point along X in the grid and looks at the propagation of an error at that point, making certain reasonable assumptions, using Fourier series (which is why the method on occasion is also called the Fourier series method).

We need first to develop an argument that allows us to separate concentrations and errors. Let the vector $\boldsymbol{C}$ of concentrations along a space coordinate X be the sum of a vector $\hat{\boldsymbol{C}}$ of exact values, with an error vector $\boldsymbol{\epsilon}$ added:

$$\boldsymbol{C} = \hat{\boldsymbol{C}} + \boldsymbol{\epsilon} \,. \tag{14.5}$$

We are interested in what happens to the errors, and the linear nature of (14.5) allows us to subtract the concentrations out of the diffusion equations,

leaving only the errors. These have simpler boundary conditions; for example, errors far away from the electrode are zero. This simplifies the form of the equations describing the changes in errors. The diffusion equation for the errors is then

$$\frac{\partial \epsilon}{\partial T} = \frac{\partial^2 \epsilon}{\partial X^2} \tag{14.6}$$

with either a Dirichlet or derivative (Neumann) boundary condition at the electrode, and $\epsilon(X \to \infty) = 0$. The equation is discretised, using equal intervals, as usual, at the point with index i. For example, using method EX, we have

$$\epsilon_i' = \epsilon_i + \lambda \left(\epsilon_{i-1} - 2\epsilon_i + \epsilon_{i+1}\right) \ . \tag{14.7}$$

Two substitutions are now made. Firstly, it is assumed that at a given point, the value of ϵ there changes with time in a general exponential manner,

$$\epsilon = \epsilon_0 \, \mathrm{e}^{\alpha T} \tag{14.8}$$

in which α is some complex constant and ϵ_0 is an initial value. We then set, for convenience $\exp(\alpha) = \xi$, so that we can substitute for ϵ', as in

$$\epsilon(T + \delta T) = \xi \, \epsilon(T) \ . \tag{14.9}$$

It is seen that ξ is an amplification or propagation factor, and the object of the exercise is now to find out under what conditions its magnitude is less than unity, which is the stability condition.

The other substitution is for the errors, which are expressed in terms of a Fourier series along the space coordinate (N points along X):

$$\epsilon_i = \sum_{n=0}^{N} a_n \mathrm{e}^{j\beta_n X_i} \ , \tag{14.10}$$

where $X_i = iH, i = 1 \ldots N$ and β_n are "frequencies" as inverse distances along X, j here being the imaginary number $\sqrt{-1}$. The coefficients a_n are the spectral intensities and are not known. A given error is thus a sum of a number of Fourier components, and we can limit our view to any one of the components. Writing now simply β for any one of the range of frequency values, and substituting both (14.9) and (14.10) in (14.7), we have for the point at index i, going from time step k to $k+1$,

$$\xi^{k+1} \mathrm{e}^{j\beta iH} = \xi^k \mathrm{e}^{j\beta iH} + \lambda \xi^k \left(\mathrm{e}^{j\beta(i-1)H} - 2\mathrm{e}^{j\beta iH} + \mathrm{e}^{j\beta(i+1)H}\right) \tag{14.11}$$

(taking the common ξ^k term on the right-hand side outside the bracket). Division by ξ^k and $\mathrm{e}^{j\beta iH}$ produces

$$\xi = 1 + \lambda \left(\mathrm{e}^{-\beta H} - 2 + \mathrm{e}^{\beta H}\right) \tag{14.12}$$

leading to

$$\xi = 1 - 4\lambda \sin^2 \left(\frac{i\beta}{2} \right) \ . \tag{14.13}$$

Since the maximum value for the $\sin^2$ term is unity, this equation leads to the condition for $|\xi| \leq 1$, that $\lambda \leq \frac{1}{2}$. This is the well known λ limit for the method EX. EX is conditionally stable.

Repeating this for the method BI, where the discrete equation at point i is

$$\epsilon'_i = \epsilon_i = \lambda \left(\epsilon'_{i-1} - 2\epsilon'_i + \epsilon'_{i+1} \right) \ . \tag{14.14}$$

leads to

$$\xi = \frac{1}{1 + 4\lambda \sin^2 (\frac{i\beta}{2})} \tag{14.15}$$

which satisfies the stability condition for all values of λ. What is more, the greater λ is, the closer ξ approaches zero. So this method is unconditionally stable, and L-stable.

A similar analysis for CN results in

$$\xi = \frac{1 - 2\lambda \sin^2 \left(\frac{i\beta}{2} \right)}{1 + 2\lambda \sin^2 \left(\frac{i\beta}{2} \right)} \tag{14.16}$$

which also sets no limits on λ, fulfilling the condition. In this case, however, as λ increases, $\xi \to -1$, which explains the oscillatory behaviour of the method CN. It is unconditionally but A-stable.

This sort of analysis can be applied to other methods. Britz and Strutwolf [152] applied it to the BDF method using 5-point discretisation along X, and, also for 5-point approximations, Strutwolf and Britz [531] applied it to extrapolation. For a multilevel method such as BDF, the analysis results in a polynomial in ξ, and complex roots are possible. For example, Lapidus and Pinder [350] treat the DuFort-Frankel method; it results in a quadratic equation in ξ but it is clear that is is unconditionally stable (even though we have seen that is not consistent).

14.3.3 Matrix Stability Analysis

The von Neumann method described above usually works well, and is reasonably easy to apply. One reason it works well, despite the fact that it totally ignores conditions at the boundaries, is that errors that often arise at interior points away from the boundaries and spread from there [private communication with O. Østerby 1996]. However, boundary conditions can affect stability, especially if derivative (or mixed) boundary conditions hold [116, 117, 118, 119, 334]. It might be safer to consider all points in space in some way. The following somewhat brief treatment is described in greater mathematical detail in such texts as Smith [514] or Lapidus and Pinder [350].

Eigenvalue Method

Equation (14.6), when discretised for, say, a potential jump experiment (Cottrell), gives rise to a system of *ode*s, depending on the method of discretisation used. For example, using the EX method, we have for the ith equation

$$\epsilon_i' = \epsilon_i + \lambda\big(\epsilon_{i-1} - 2\epsilon_i + \epsilon_{i+1}\big) \tag{14.17}$$

and recalling that at the electrode and the outer boundary in the bulk, ϵ is zero, we can write the whole system (14.17) in the form

$$\boldsymbol{\epsilon}' - \boldsymbol{\epsilon} = \lambda \boldsymbol{A}\boldsymbol{\epsilon} \tag{14.18}$$

with concentrations (or errors) now represented as vectors, and $\boldsymbol{A}$ being the matrix of coefficients, in this case given by

$$\boldsymbol{A} \equiv \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} . \tag{14.19}$$

The equation can be rearranged explicitly for $\boldsymbol{\epsilon}'$,

$$\boldsymbol{\epsilon}' = \boldsymbol{P}\boldsymbol{\epsilon} \tag{14.20}$$

with

$$\boldsymbol{P} = [\boldsymbol{I} + \lambda \boldsymbol{A}] . \tag{14.21}$$

We now see that one way of describing the simulation is as a series of multiplications of the error vector with the propagation matrix $\boldsymbol{P}$. The matrix method of stability analysis focusses on $\boldsymbol{P}$ and its effect on the whole error vector. That vector must not grow without limit, and to ensure this, there are some related conditions. One of them is that the magnitude of the largest eigenvalue of $\boldsymbol{P}$ must not exceed unity. In fact, in this particular case (see Smith [514] or any similar text), the eigenvalues are known, and this leads again to the condition on λ, that is, $\lambda \leq 0.5$. Not all cases of simulation methods lead to propagation matrices whose eigenvalues are known, and in these cases, they must be found numerically. They can be complex, as is the case, for example, for methods like BDF using 3 or more points. Some examples are now given.

One convenient way to illustrate stability is to plot the eigenvalue of maximum magnitude in the complex plane for a number of λ values. This provides the so-called spectral radius of the method. In the following examples, a value of $N = 20$ was used throughout. The actual number of spatial points makes little difference to the look of the diagrams to follow, and the chosen value was small enough for short computation times. Generally, the range of λ was

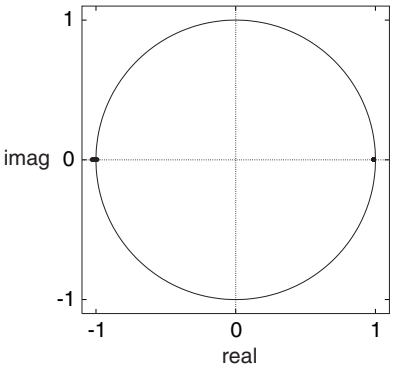


Fig. 14.6. Maximum absolute eigenvalues for EX for $0.49 \leq \lambda \leq 0.51$ ($N = 20$)

chosen as $10^{-6} \leq \lambda \leq 10^{6}$, in a geometric sequence, except for method EX, where the range was narrowed to around the stability limit, $0.49 \leq \lambda \leq 0.51$. Consider Fig. 14.6. Maximum-magnitude eigenvalues were calculated for matrix $\boldsymbol{P}$ as in (14.21). The points on the right of the figure are those for λ up to (and including) 0.5. For greater values, the eigenvalues jump to the negative real end and fall outside the unity circle. That is, in the parlance of numerical analysis, the spectral radius exceeds unity. This means that errors will increase in magnitude, and also oscillate as they in fact do. This is exactly what one sees in practice, where increasing oscillations in concentrations are observed.

A method known to be stable is BI. The discretisation for the errors is

$$\epsilon'_i - \epsilon_i = \lambda\big(\epsilon'_{i-1} - 2\epsilon'_i + \epsilon'_{i+1}\big) \tag{14.22}$$

which forms the system

$$[\boldsymbol{I} - \lambda \boldsymbol{A}]\boldsymbol{\epsilon}' = \boldsymbol{P}\boldsymbol{\epsilon} \tag{14.23}$$

leading to the propagation matrix

$$\boldsymbol{P} = [\boldsymbol{I} - \lambda \boldsymbol{A}]^{-1} \,. \tag{14.24}$$

Figure 14.7 shows the spectral radii for this, for the range $10^{-6} \leq \lambda \leq 10^{6}$, in a logarithmic sequence. For increasing λ the points move from right to left in the figure. All eigenvalues are real, and all fall inside the unit circle, confirming the stability of BI. One also finds that the eigenvalues approach zero with increasing λ, so that for high λ values errors will be damped more effectively.

Another method worth considering is CN, and Fig. 14.8 shows the result. Here, discretisation is

$$\epsilon'_i - \epsilon_i = \frac{\lambda}{2}\big(\epsilon'_{i-1} - 2\epsilon'_i + \epsilon'_{i+1}\big) + \frac{\lambda}{2}\big(\epsilon_{i-1} - 2\epsilon_i + \epsilon_{i+1}\big) \tag{14.25}$$

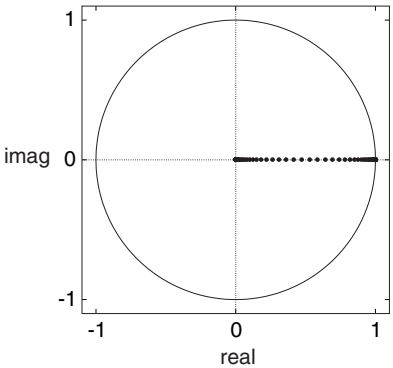


Fig. 14.7. Maximum absolute eigenvalues for BI for $10^{-6} \leq \lambda \leq 10^{6}$ ($N = 20$)

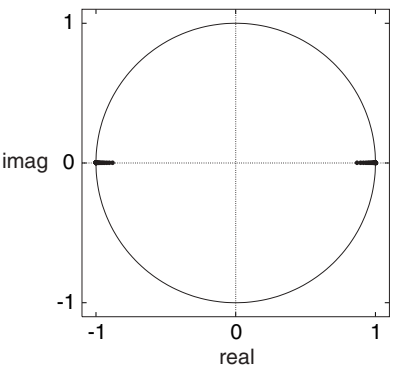


Fig. 14.8. Maximum absolute eigenvalues for CN for $10^{-6} \leq \lambda \leq 10^{6}$ ($N = 20$)

which forms the system

$$\left[\boldsymbol{I} - \frac{\lambda}{2}\boldsymbol{A}\right]\boldsymbol{C}' = \left[\boldsymbol{I} + \frac{\lambda}{2}\boldsymbol{A}\right]\boldsymbol{C} \tag{14.26}$$

leading to the propagation matrix

$$\boldsymbol{P} = \left[\boldsymbol{I} - \frac{\lambda}{2}\boldsymbol{A}\right]^{-1}\left[\boldsymbol{I} + \frac{\lambda}{2}\boldsymbol{A}\right] \ . \tag{14.27}$$

The figure shows that again, for small λ values, the points move from the right, being close to unity, towards the left. Unlike with BI, however, they do not approach zero but at some λ value, depending on N, they cross to the negative side and approach -1. This underlines the oscillatory behaviour of CN, especially for large λ.

The methods considered above all show purely real eigenvalues. One method for which they are complex is BDF. For 3-point BDF, the discretisation is

$$'\epsilon_i - 4\epsilon_i + 3\epsilon'_i = 2\lambda\left(\epsilon'_{i-1} - 2\epsilon'_i + \epsilon'_{i+1}\right) \tag{14.28}$$

(recalling that $'\epsilon$ stands for $\epsilon(T - \delta T)$). This is a slightly more complex situation. The system can be written in vector-matrix form as

$$[3\boldsymbol{I} - 2\lambda\boldsymbol{A}]\boldsymbol{\epsilon}' = 4\boldsymbol{\epsilon} - '\boldsymbol{\epsilon} \tag{14.29}$$

and clearly, propagation is now not simply a matter of multiplying a single vector by a propagation matrix; we have two old vectors to deal with, $'\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}$. First we write (14.29) in the usual form,

$$\boldsymbol{C}' = [3\boldsymbol{I} - 2\lambda\boldsymbol{A}]^{-1}(4\boldsymbol{C} - '\boldsymbol{C})\,, \tag{14.30}$$

which we write, letting $\boldsymbol{B} = [3\boldsymbol{I} - 2\lambda\boldsymbol{A}]^{-1}$, as

$$\boldsymbol{C}' = \boldsymbol{B}(4\boldsymbol{C} - '\boldsymbol{C})\,. \tag{14.31}$$

This will be needed later. We now need another equation. First, consider that when using BDF, we calculate the new vector from the two old ones, and the previous present one becomes the previous older one. This can be expressed as a vector of vectors operation. Let the new double-length vector $\boldsymbol{e} \equiv [\boldsymbol{\epsilon} \quad '\boldsymbol{\epsilon}]^T$, consisting of the two known vectors, and the new one, $\boldsymbol{e}' \equiv [\boldsymbol{\epsilon}' \quad \boldsymbol{\epsilon}]^T$, the present one and the new one, to be calculated. To express this mathematically as a propagation from $\boldsymbol{e}$ to $\boldsymbol{e}'$, we need one more equation, and augment system (14.30) to the system of systems

$$\begin{aligned} \boldsymbol{e}' &= \boldsymbol{B}(4\boldsymbol{e} - '\boldsymbol{e}) \\ \boldsymbol{e} &= \boldsymbol{e}\,. \end{aligned} \tag{14.32}$$

The seemingly redundant second equation now enables us to write this in matrices-in-matrix form:

$$\left[\boldsymbol{e}'\right] = \begin{bmatrix} 4\boldsymbol{B} & -\boldsymbol{B} \\ \boldsymbol{I} & 0 \end{bmatrix} \boldsymbol{e} \tag{14.33}$$

where clearly now the propagation matrix $\boldsymbol{P}$ is the 2×2 matrix of matrices. This can be expanded into a larger $2N \times 2N$ matrix, and eigenvalues computed. Figure 14.9 shows the result. As with BI, the eigenvalues for small λ are at the right of the figure, close to unity and real. At some λ value, depending on N, the eigenvalues become complex and pair up into complex congugates, following, with increasing λ, a circular path towards the centre of the circle. So, as for BI, high λ means effective error damping, but the damping factor is complex. Complex propagation factors imply the possibility of oscillations with periods perhaps of several time intervals. For higher-point BDF variants, the points do not follow a circular path, but pairs of arcs with increasing magnitude, with some values in the left-hand plane; and eventually, for more than 7 points, reaching outside the unit circle, indicating instability [320].

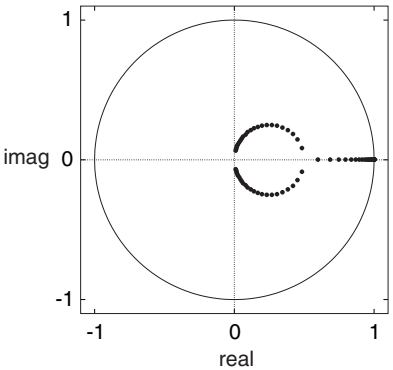


Fig. 14.9. Maximum absolute eigenvalues for 3-point BDF for $10^{-6} \leq \lambda \leq 10^{6}$ ($N = 20$)

This illustrates how diagrams of spectral radii can provide information on the stability of a given method.

So far, the analysis has been for the Cottrell method. It is of interest to see how it changes for derivative boundary conditions. This only changes a single line in the discrete system of equations. For constant current, for example, we might use a two-point expression

$$\epsilon_1 - \epsilon_0 = 0 \tag{14.34}$$

(there is no current for errors at the electrode) or

$$\epsilon_0 = \epsilon_1 \tag{14.35}$$

and this changes the first equation for EX to

$$\epsilon_1' = \epsilon_1 + \lambda(\epsilon_1 - 2\epsilon_1 + \epsilon_2) \tag{14.36}$$

which makes the coresponding coefficient matrix

$$\boldsymbol{A} \equiv \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} . \tag{14.37}$$

The analysis is otherwise the same, but the effects are not, as will be discussed below. Extension to a greater number of points to express the derivative is obvious.

Matrix Norm Method

A different but mathematically equivalent way to perform the matrix analysis is to use matrix norms rather than eigenvalues. Smith [514] states that the condition for all eigenvalues not to exceed unity in magnitude is equivalent to the demand that the infinity norm of the propagation matrix $\boldsymbol{P}$ does not exceed unity. That norm can be more easily computed than eigenvalues. In fact, all norm definitions can be used for this purpose. The infinity norm of a matrix is defined as the largest of the sums of the moduli of the row values. So this is another criterion for stability. For a method such as BI, we find that for all λ values, the norms are a little less than unity. For CN and BDF3, however, the norms can be greater than unity and this might cause concern. However, a slightly more relaxed condition for stability is that the norms of increasing multiples of the propagation matrix do not increase indefinitely. The limit of this series of norms may be greater than unity, but it must exist. If this condition holds, then errors will not build up indefinitely, and the method is stable. Such norms exceeding unity have been found for certain electrochemical simulations [119], for CN under mixed boundary conditions. So, although norms are easier (and faster) to compute than eigenvalues, their use is not as straight-forward as the use of eigenvalues.

14.3.4 Some Special Cases

There are some special conditions in electrochemical simulations that have an effect on stability.

Bieniasz and Britz [94,112] investigated the effect of the presence and rate of homogeneous chemical reaction (hcr) accompanying electron transfer and found some effects. Writing $\xi = K\delta T$ (K being the dimensionless first-order hcr rate constant as in Chap. 5, (5.12)), Bieniasz found [94] that for method EX, using the parallel discretisation as in (5.13), the upper permissible λ value for stability decreases linearly with ξ, and reaches zero for $\xi = 2$. For larger ξ values and this discretisation, EX is always unstable. For the sequential discretisation (see Sect. 5.4, page 77 and Appendix B, page 289), in which the chemical reaction is allowed to act on concentrations that have undergone diffusional changes, the situation is more complex but the same upper limit on ξ holds. A similar stability decrease and the same limit on ξ were found for explicit RK. Stable methods such as CN or Saul'yev were unaffected by an hcr in the sense that they remained stable for all ξ values. These findings were then confirmed numerically [148].

Another factor affecting stability is a derivative boundary condition. Keast and Mitchell pointed out potential problems with CN in this regard [334], and investigations in the electrochemical context revealed some problems with methods otherwise thought to be unconditionally stable, such as CN and Saul'yev [116, 117, 118]. The CN method was found to become

unstable for $\lambda > 4$ if a heterogeneous rate constant setting a derivative boundary condition decreased with time [116], as happens, for example, in linear sweep experiments on a quasireversible system. Also, certain (unusual) ways to discretise the derivative boundary conditions can render CN unstable [119]. Fortunately, these effects are seen under somewhat extreme conditions; indeed they are difficult to demonstrate numerically, so they might be considered of only academic interest.

14.4 The Stability Function

Another way of investigating stability, that at the same time provides information on the behaviour of a given method, is what Gourlay and Morris [277] call the symbol of the algorithm, also called the symbol of the method [514] or, more logically perhaps, the stability function [286]. It is developed from Padé approximations to the general solution of the diffusion equation. Equation (14.6) can be semidiscretised to the system of *ode*s as

$$\frac{d\boldsymbol{\epsilon}}{dT} = \frac{1}{H^2}\boldsymbol{A}\boldsymbol{\epsilon} \tag{14.38}$$

with $\boldsymbol{A}$ defined in (14.19) for the Cottrell experiment. At this stage, a particular simulation method, that is, a time-integration algorithm, has not been specified. The general solution of this, analogously with the plain *ode* $y' = -ay$ is

$$\boldsymbol{\epsilon}(T) = \exp\left(\tfrac{1}{H^2}\boldsymbol{A}T\right) \tag{14.39}$$

where the exponential is understood as its series definition,

$$\exp(\boldsymbol{A}) = \boldsymbol{I} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \frac{\boldsymbol{A}^3}{3!} + \dots \ . \tag{14.40}$$

This leads, for a given step of length δT, to the solution

$$\boldsymbol{\epsilon}' = \exp(\lambda\boldsymbol{A})\boldsymbol{\epsilon} \ . \tag{14.41}$$

It is at this point, that the various Padé approximations to the exponential functions come in. One of them is simply the sequence on the right-hand side of (14.40) cut off after the first two terms, producing the propagation equation

$$\boldsymbol{\epsilon}' = (\boldsymbol{I} + \lambda\boldsymbol{A})\epsilon \tag{14.42}$$

which is clearly the method EX, and the truncated series is the (0,1) Padé approximation. Smith [514] tabulates a number of these approximations. For example, the (1,0) variant generates

$$\boldsymbol{\epsilon}' = \frac{1}{(\boldsymbol{I} - \lambda\boldsymbol{A})}\boldsymbol{\epsilon} \tag{14.43}$$

which is seen to be the BI method. There is a number of approximants, the (1,1) one leading to CN. The various algorithms have in this way been unified under one system.

The next step is to consider the eigenvalues of $\boldsymbol{A}$ in the propagation formulae such as (14.42) and (14.43). These are all negative, and we set $z = \lambda\omega$, where ω are the eigenvalues. Note that Hairer and Wanner use the negative of z for their stability functions [286]. Since λ can have any value greater than zero, we can dissect a particular propagation equation in terms of z. Thus, for method EX, we have for the "stability function" $R(z)$,

$$R(z) = 1 - z \tag{14.44}$$

which decreases linearly with z and goes below -1 for $z > 2$. For the actual maximum-magnitude eigenvalue equal to -4 (which is the case), this goes back to the known stability criterion $\lambda \leq 0.5$.

For BI, (14.43) becomes the stability function

$$R(z) = \frac{1}{1+z} \tag{14.45}$$

and for CN (using the (1,1) Padé form [514]), it is

$$R(z) = \frac{1 - \frac{z}{2}}{1 + \frac{z}{2}} \,. \tag{14.46}$$

An interesting case is extrapolation, and we consider the simplest variant, the second-order case. It is a number of steps using BI. First one takes two successive steps with half step size, and subtracts from twice the result of this the result of one whole step. Thus, we can directly write

$$R(z) = 2\left(\frac{1}{1+\frac{z}{2}}\right)^2 - \frac{1}{1+z} \tag{14.47}$$

which expands to

$$R(z) = \frac{1 + z + \frac{1}{4}z^2}{1 + 2z + \frac{5}{4}z^2 + \frac{1}{4}z^3} \,. \tag{14.48}$$

Figure 14.10 shows the result for the three methods BI, CN and second order extrapolation. Note first the smooth but rather slow decline to zero of BI. Note next that CN crosses zero at some z, and therefore, some λ, and approaches -1 for large arguments. This illustrates what we know about CN, that is it becomes more and more oscillatory for large λ. Finally, second-order extrapolation also crosses zero, but then approaches zero again, and of the three methods, it converges most rapidly to zero. This is an attractive feature. It remains to be seen, however, how the method compares with others in terms of efficiency.

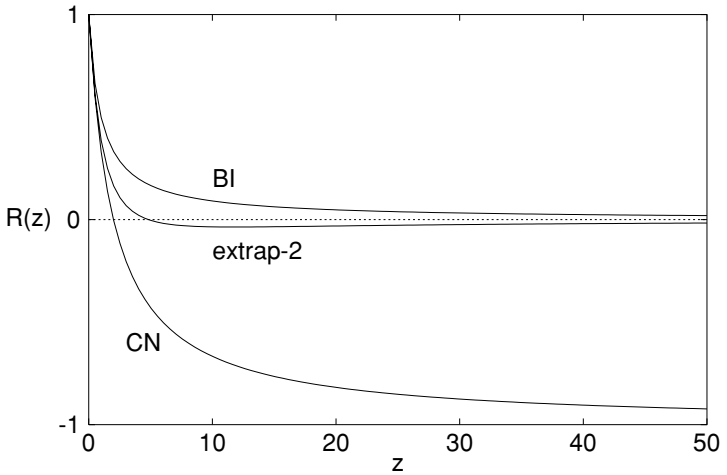


Fig. 14.10. Stability function $R(z)$ for the marked methods

14.5 Accuracy Order

In other chapters, the **order** of accuracy of various methods is referred to. Here this concept is defined and two methods of calculating the order are presented.

A simulation results in a number (or a vector of numbers) at some time. Depending on the dimensionality of the problem, the simulation uses intervals in time δT and one or more space intervals. Often there is only one space interval, here given the symbol H. A result – a current, or a concentration, for example – will, due to truncation errors, have an error associated with it, that can be expressed in the following way. The discussion is, for the moment, restricted to an *ode* with interval size h. Then the simulated result at time t can be written as a polynomial

$$u = \hat{u} + a\,h + b\,h^2 + \ldots \tag{14.49}$$

where $\hat{u}$ is the underlying true solution (known or not, see below), and $a, b, \ldots$ are constants. For a given method, the polynomial will be dominated by a certain power term. Let this term be the power p, so that the above equation can be written as

$$u = \hat{u} + O(h^p)\ . \tag{14.50}$$

The number p is of great interest, more so than the constants, which are generally unknown and are usually unimportant (except in rare cases) when deciding on a given method. This is because a high order accuracy means that if we decrease h, we dramatically improve the accuracy. Conversely, this is not the case for a small p. So, first-order methods such as EX or BI mean that we must decrease the intervals greatly in order to achieve some

target accuracy, implying perhaps unacceptable computing time. However, as discussed in earlier chapters, this is not always the only criterion; a given high-order method might require so much computing time, that a lower-order method is preferable.

14.5.1 Order Determination

There are two ways of determining the order of a method, depending upon whether we have an exact solution to compare with, or not.

If we know an exact solution, Method 1 is used. First a result u_1 is calculated, using interval size h. From (14.49), we have

$$u_1 = \hat{u} + a\,h + b\,h^2 + \ldots \; ; \tag{14.51}$$

the simulation is then repeated at a new interval size, αh, so that the new result is

$$u_2 = \hat{u} + a\alpha\,h + b\alpha^2\,h^2 + \ldots \quad . \tag{14.52}$$

Usually, α is chosen as 2. We can now calculate errors associated with the approximate solutions,

$$e_1 = u_1 - \hat{u} = a\,h + b\,h^2 + \ldots \tag{14.53}$$

and similarly for e_2. Dividing e_2 by e_1, we have

$$\frac{e_2}{e_1} = \frac{a\alpha\,h + b\alpha^2\,h^2 + \ldots}{a\,h + b\,h^2 + \ldots} \; . \tag{14.54}$$

If the dominant term is that with h, the ones in higher powers can be dropped and we get

$$\frac{e_2}{e_1} \approx \alpha \; . \tag{14.55}$$

If the dominant term is that in h^2, all others are dropped and we get

$$\frac{e_2}{e_1} \approx \alpha^2 \; . \tag{14.56}$$

The order, expressed as the power in h, is then equal to $\log_\alpha(\frac{e_2}{e_1})$. As mentioned above, $\alpha = 2$ is usually chosen for convenience.

If we do not know an exact solution, we can still estimate the error order by Method 2, as described by Østerby [430]. We must use one more interval size, $\alpha^2 h$. We then have a third result:

$$u_3 = \hat{u} + a^2\alpha\,h + b\alpha^4\,h^2 + \ldots \quad . \tag{14.57}$$

Instead of using errors (we do not know them), we use

$$\frac{u_3 - u_2}{u_2 - u_1} = \frac{\alpha(\alpha-1)\,ah + \alpha^2(\alpha^2-1)\,bh^2}{(\alpha-1)\,ah + (\alpha^2-1)\,bh^2} \tag{14.58}$$

(which eliminates the errors) and again, we obtain the factor α for a first-order result and α^2 for a second-order result, etc. The unknown exact solution is subtracted out in this process. Often, this method yields less clear results, particularly if the actual errors are large, than the first method requiring only two simulations.

In the preceding, only a single interval has been considered. In electrochemical simulations, there are at least two: in time and space; and there may be more, if the simulation is in more than one dimension. In such cases, the various intervals might affect the accuracy to different orders. Thus, both methods EX and BI have concentration accuracies of order $O(\delta T, H^2)$, while CN has $O(\delta T^2, H^2)$. It is clear now that EX and BI lack a high-order time derivative. It is also clear that the order does not tell the whole story; we know that CN has a higher accuracy order than both EX and BI, but we also know that it is oscillatory for the first so many steps. The order tells us nothing about this. When measuring the order of a method with several different interval sizes, one must obviously not only vary the one being measured, keeping the others constant; one must also ensure that the other interval sizes are rather small, so that the accuracy of the result is not limited by them. Otherwise, the order calculation will not be as clear.

One might also suspect that a higher-order method might still have large errors compared with some lower-order method, because we ignore the polynomial constants, and they are surely different for different methods. This is so but in all practical cases, the constants do not affect the accuracy as strongly as the order itself, so a higher-order method always leads to smaller errors. It might, however, do so at the expense of more computing time, and this is gone into next.

Another aspect is that of starting a given method with another. For example, one way to start BDF is to use BI for the first step. A single BI step (see Chap. 3) is second-order acccurate with respect to the time interval. The next step is BDF3 (three-point BDF), and so on. One might expect that the second-order error introduced by the first BI step can be "diluted" by the subsequent high-order steps. However, this is not so. Normally, the largest errors appear at the start of a simulation, and remain to contaminate the result. So the described procedure, the rational start, yields second-order accuracy, no matter what order BDF one uses. This is why, when using this start, one might as well use three-point BDF. Higher-order BDF forms do not improve the result, as is seen in Table 14.1. In that table, the simple *ode*, $y' = -y$ was simulated, using 100 steps of interval size 0.01. Four ways of handling BDF were used: the simple start without any time correction ("simp"); the simple start with correction by half a time interval ("simp+"); the rational start, working up from BI as described ("rat"), and lastly, adapting the Kimble and White (KW) method to provide a high-order start, as described on page 64. The orders were calculated using Method 1. Method 2 was also used and gave the same results. The actual errors are also shown. The same problem

Table 14.1. Error orders (and errors at $t = 1$) for some BDF starts, for the *ode* $y' = -y$

Start	$k = 2$ (BI)	$k = 3$	$k = 4$	$k = 5$
simp	1.00 (1.8×10^{-3})	1.01 (1.9×10^{-3})	1.00 (1.8×10^{-3})	1.00 (1.8×10^{-3})
simp+	1.99 (6.1×10^{-6})	2.01 (2.0×10^{-5})	1.98 (7.5×10^{-6})	2.00 (7.7×10^{-5})
rat	1.00 (1.8×10^{-3})	2.00 (1.5×10^{-5})	1.99 (2.3×10^{-5})	2.00 (2.5×10^{-5})
KW	$-(5.5\times10^{-3})$	1.98 (-1.2×10^{-5})	3.00 (9.1×10^{-8})	3.97 (-7.1×10^{-10})

Table 14.2. Calculated orders for the *ode* $y' = -y$ using extrapolation at various orders

Extrapolation Order	Accuracy Order (error)
2	1.99 (6.0×10^{-6})
3	2.99 (2.5×10^{-8})
4	3.97 (6.7×10^{-11})

was then simulated using extrapolation with orders $2 \ldots 4$, as described in 4.9, with the same interval size and number of steps. Table 14.2 shows the results.

In Table 14.1, there is one anomaly, for BI ($k = 2$) with the time correction. This is not expected to be a good method but turned out second-order accurate with a comparatively very small error. This must be regarded as fortuitous, due to time-varying time shifts [140] and perhaps to the particular *ode* used as test case. The other results are as expected. Note the order 2 for all rational starts, and the impressive high orders for KW, and corresponding increasingly smaller errors, for the KW start. Note also that the calculated orders are not exact integers, which is due to some uncertainties, or due to terms other than the dominant order terms playing a small part. Table 14.2 also shows expected results, and shows that extrapolation at high order results in conveniently small errors.

14.6 Accuracy, Efficiency and Choice

We come now to the choice of method. There are no hard and fast rules here, the final choice depending to a large extent on personal preference and the inclination towards programming. Computers are now so fast that all but hard simulation problems such as CVs of, say, 2D problems or 3D problems execute in a very short time – usually just a few seconds. Here, the main bottleneck will be the programming itself, including finding the initial programming errors. If a given method results in a savings of a second or so, it might not be worth the extra effort in terms of paper work and programming. Nevertheless, some rough guidelines will be provided here.

First of all, we must be clear about what we mean by accuracy, or what we wish to be accurate. Usually, the result of a simulation is a current function of time, so one argument might be, that we do not care about concentrations, whereas the current must be accurate [481]. There is the possibility of error cancelling, which can lead to small errors in a current, whereas the concentration errors are larger (in a relative sense). This may not be reliable for all parameter choices. Empirical adjustments have also been practiced. Thus, Feldberg in his seminal chapter [229] introduced the doubtful device of subtracting half a time interval from all t values in a simulation using the box method (EX). Although there was – and is – no justification for this, it seemed to give better results. The practice has been followed by others, for example in the text of Bard and Faulkner [74], as if by tradition. If one uses better methods than explicit, then it becomes clear that this device is not appropriate – except in the case of BDF, using the simple start. In their original work on the BDF method, Mocak and Feldberg advocated the subtraction of half a time interval. This was examined in some detail [140,154], and it was found that it indeed leads to a dramatic improvement in accuracy (see also Table 14.1). A justification for the device was then shown [142,155]. Since this device is reasonable only for this particular case, this is a remarkable coincidence, and the only case in which this device is in fact reasonable. For other cases such as method EX the device must be regarded as an empirical adjustment, and positive results resulting from it as fortuitous.

The numerical solution produces concentration values, and one must therefore strive to obtain as accurate values for these as possible, so that currents calculated from them might also be accurate. For this reason, Bieniasz now makes a practise of showing errors across the whole concentration profile, when reporting a new simulation method [100,106,110], or at least a few samples from the profile [109].

Accuracy alone is not a sufficient criterion for a good method, however. One can, for example, usually drive method EX to any target accuracy, by simply refining the time and spatial intervals sufficiently. The result may then require excessive computer time, although not always. The target accuracy itself is subject to discussion. It must be kept in mind that the computed currents are normally compared with experimentally measured values, in order to obtain some experimental parameters. The measurements can rarely be carried out with better than 1% accuracy, because they rely on components such as resistors etc. which have, at best, that level of accuracy. So one might set a target accuracy at a relative 10^{-3}. Then, one measures the computing time needed to achieve that target accuracy. The most efficient method is then the one using the least computing time. This will rarely be the method that provides the best accuracy for a given space/time grid. For example, the KW start for BDF (see Table 14.1 and Sect. 4.10.1) clearly provides a high-order start and results in impressively small errors for the higher-order BDF variants. However, applied to electrochemical digital simulation, it was

found to be inefficient [154], and it was found that for BDF, the most efficient method is the simple start with subsequent subtraction of half a time interval, as mentioned above. Thus, we seek efficient methods that minimise computing time for a target accuracy.

Another factor is programming effort, including planning on paper. The KW start mentioned above is a case in point. It is not trivial to implement for a *pde* in a computer program, and we might consider ourselves lucky that it was shown to be inefficient. The same might be thought of the Rosenbrock methods, which are less easy to program than, say, BDF or the extrapolation variants, and also for OC. This is to a large extent subjective. Someone who has worked with these harder methods such as Rosenbrock and OC, might not believe that they are hard to program, and will then achieve good execution efficiency from them, whereas others will prefer easier methods. Here, the discussion will be restricted to a chosen few methods, regarded, by the present author, as good compromises between programming effort and efficiency.

CN is not among this group. As described earlier in Chap. 8, CN leads to initial oscillations. These can be damped by several devices [149, 432], but these are either not very effective, or demand increased programming effort. Thus, the Pearson method [437], in which the first step is simply subdivided into a number of equal substeps such that each one corresponds to a λ value of around unity, will obviously mean very many substeps for large λ, although it is easy to implement from a programming point of view. Less easy is the use of unequal substeps, but here some experience is necessary in order to choose the most suitable expansion factor [149] (and may necessitate an LU decomposition at every substep). Lastly, the effective device of taking one or more steps using BI, then following with CN as suggested [149, 461], necessitates the addition of the BI method to a given CN program. Some might consider this too much effort. Given that there are other implicit methods, as easy to program as CN, that are also as efficient, we can exclude CN from our menu of choices.

The two methods that stand out in terms of efficiency and convenience are BDF and extrapolation. Both require minimal programming effort, and can be extended to higher-order spatial derivatives. However, in the case of BDF, a limit is encountered. For the most convenient start-up methods such as the simple or the rational start, the accuracy from BDF is limited to $O(\delta T^2)$. This means for one thing that one need not go beyond 3-point BDF (which is $O(\delta T^2)$ in itself), but that no marked improvement can be gained from higher-order spatial derivative approximations, because there will then be a mismatch between the accuracy orders with respect to the time and spatial intervals.

Extrapolation does require extra concentration arrays (as does BDF) leading to less convenient programming, but higher-order extrapolation can rather easily be achieved, and thus the use of higher-order spatial derivative approximations can be used to advantage.

It has been found by numerical experiment [154] that these two methods are about the most efficient, when combined (especially in the case of extrapolation) with higher-order spatial derivative approximations. The variants found to be best are 3-point BDF using the simple start and with subtraction of half a time interval, and extrapolation, possibly with the higher spatial derivative orders. A single example of a comparison is now presented. The Cottrell system was simulated, using unequal intervals with 40 points and a smallest spatial interval equal to 0.01. This gives an expansion factor γ of 1.11, quite moderate. Two simulation methods were chosen. Second-order extrapolation was chosen, being quite simple to program, as well as second-order BDF, using the simple start with half time-interval correction. The simulation was run with various time intervals (expressed as N_T, the number of steps to $T = 1$), and the error in the current at that time measured, as a three-point approximation on the nonuniform grid. The second spatial derivative was approximated by a four-point formula, which allows the use of a modified Thomas algorithm, see Sect. 8.4 on page 124. In the figures, this is given as the logarithm of the relative absolute error. That is, if the simulated current be i_{sim} and the analytically known value be i_{anal}, then what is plotted (marked in the figures as `"log|err|"`) is the quantity $\log_{10} |\frac{i_{sim}-i_{anal}}{i_{anal}}|$. Note that (Fig. 14.11) both curves show similar accuracy at small N_T. If one measures the gradient there, it verifies the expectation of an accuracy order of $O(\delta T^2)$. However, extrapolation then shows a sharp dip, followed by an approach to a constant error. The dip is due to the actual error crossing the zero line. BDF simply flattens out and approaches the same constant error.

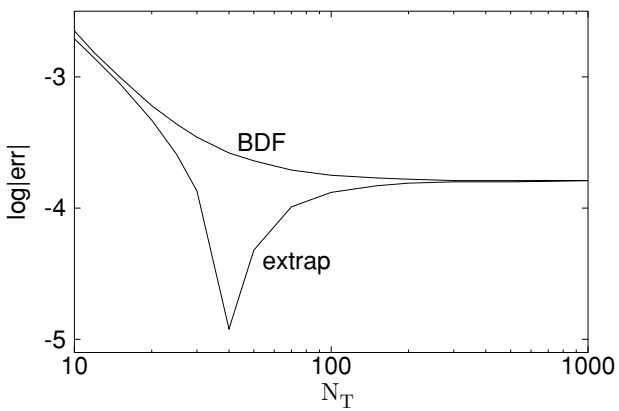


Fig. 14.11. $\log_{10} |\frac{i_{sim}-i_{anal}}{i_{anal}}|$ vs N_T for extrapolation and BDF (both 2nd-order)

This arises because with decreasing time intervals per step, the error from the second spatial derivative approximation makes itself felt more and more. The curve for BDF does not show the dip. Looking at the two curves, one might consider both methods as roughly equal, which in fact they are in terms of accuracy. However, what interests us is the computing time (cpu) used. This was also measured. Figure 14.12 shows that the times are very short, measured in milliseconds, so they were measured by letting the working parts of the programs execute a sufficient number of times, so that an accurate time could be measured. Now a difference is noted: BDF uses about half as much cpu time for a given step size. If we choose some target accuracy, for example 10^{-3} or -3 on the log-scale, then BDF reaches this target at about half the cpu time of that for extrapolation. Therefore, BDF would here be chosen.

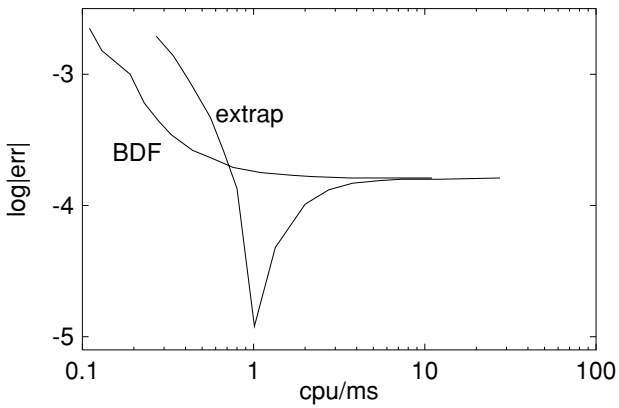


Fig. 14.12. $log_{10}\left(\left|\frac{i_{sim}-i_{anal}}{i_{anal}}\right|\right)$ vs cpu time for extrapolation and BDF (both 2nd-order)

14.7 Summary of Methods

Here, a brief summary is given of all those methods that might be of interest, with their advantages and disadvantages, as seen by the present author. References are not given, as they are provided in the sections of the book referred to.

- **EX** Chap. 5. Limited stability but accurate only to $O(\delta T)$. Easy to program, but inefficient.
- **Explicit RK**: Chap. 4, Sect. 4.5. Limited stability, marginally more efficient than EX. Easy to program.
- **BI**: Chap. 4, Sect. 4.6 and Chap. 8, Sect. 8.1. Unconditionally and L-stable but has, like EX, an accuracy of only $O(\delta T)$. Has a smooth response to initial transients. Can be valuable as a first step before 3-point

BDF (p. 58) or as one or more first steps before CN to damp oscillations (p. 121), and is the basis for higher-order methods such as extrapolation and BDF.

- **CN**: Chap. 8, Sect. 8.2. Has an accuracy of $O(\delta T^2, H^2)$, and is unconditionally but A-stable. Oscillates, however, with initial transients. With large λ, these oscillations persist over many steps, rendering CN useless. The oscillations can be damped (Sect. 8.5.1) but this defeats to some extent the simplicity of the method. For LSV or chronopotentiometry simulations, however, where no sharp initial transients occur, this might be a good method.
- **BDF**: Chaps. 4 and 8. L-stable and (for small number of levels) non-oscillatory method that can be driven to higher orders with respect to the time interval. Realistic starting strategies reduce the order to $O(\delta T^2)$, so the three-point variant is recommended, using the "simple" start with subsequent subtraction of half a time interval (Sect. 4.8.1), which in this case is justified. Reasonably easy to program.
- **Extrapolation** Chap. 4, Sect. 4.9 and Chap. 8, Sect. 8.5.2. Like BDF, is based on BI driven to a higher order. Up to $O(\delta T^4)$ is feasible to program, and there are no starting problems. Extra steps are however required, and the preferred second-order method is not as efficient as second-order BDF.
- **Unequal intervals** Chap. 7. Essential for most programs. The second spatial derivative requires four points if second order is wanted (and is recommended). With four-point discretisation, an efficient extended Thomas algorithm can be used, obviating the need for a sparse solver. Very few points can then be used across the concentration profile. Direct discretisation on the unequally spaced grid was shown to be better than using transformation and discretisation in transformed space.
- **Hopscotch** Chap. 9, Sect. 9.2.5. Stable and explicit but has the problem of "propagational inadequacy", so the ability to use large λ values is lost. Nevertheless, hopscotch continues to be used.
- **Rosenbrock** Chap. 4, Sect. 4.12 and Chap. 9, Sect. 9.4. Stable implicit RK method, and using ROS2 or ROWDA3, it can be computed explicitly. ROWDA3 is exceptionally accurate at $O(\delta T^3)$ and is especially useful for nonlinear boundary conditions (e.g. page 170), where it obviates the need for Newton iterations. However, requires programming skill and might be suitable for more professional simulation packages.
- **Higher-order methods** Chap. 9, Sect. 9.2.2 for multipoint discretisations. The four-point variant with unequal intervals is probably optimal; the system can be solved using an extended Thomas algorithm without difficulty. Numerov methods (Sect. 9.2.7) can achieve higher orders with only three-point approximations to the spatial second derivative. They are not trivial to program.
- **DuFort-Frankel** Chap. 9, Sect. 9.2.3. Stable and explicit but inconsistent and shares the propagational inadequacy problem.

- **Saul'yev** Chap. 9, Sect. 9.2.4. Stable and is calculated explicitly. If the RL and LR variants are combined, produces results equal in accuracy and order to those from CN, including its propensity to oscillate in response to an initial transient. So like CN, this might be a good method for LSV simulations.
- **OC** Chap. 9, Sect. 9.6. Produces impressively accurate results with only a few spatial points. It can be regarded as a kind of MOL, and although practitioners tend to propose advanced techniques for solving the set of *ode*s, other methods can be used, such as a Rosenbrock method or BDF. The method is not used as much as one might expect.
- **MOL/DAE** Chap. 9, Sect. 9.3. A set of *pde*s is only spatially discretised, leaving a set of *ode*s to solve. Adding the boundary conditions then produces a set of DAEs, usually solved using a professional package such as DASSL. Seems to be more difficult to program than some other methods. It will probably find its greatest use in general simulation packages.
- **Box method** Chap. 9, Sect. 9.1. The original electrochemical simulation method. With boxes, most of the above techniques can be applied. There is an unresolved issue of whether this method is inherently better than the point method, or not.
- **FEM and the like** Chap. 9, Sect. 9.5. These are possibly the most efficient methods, can be made adaptive to changing conditions, but are not trivial to program. Usually packages are used.

15 Programming

In Appendix C selected downloadable example modules, functions, subroutines and programs are discussed. All have been tested in the form in which they appear. Nevertheless, this does not guarantee that all bugs have been removed. The word "bug" encompasses the spectrum from "cosmetic, of little consequence", through "potentially serious under certain input conditions" to "fatal". The middle of the spectrum is, of course, the region causing the programmer the greatest trouble.

15.1 Language and Style

There are, at the time of writing, two strong contenders for the choice of language for digital simulation programs: Fortran 90/95 and C++. Despite the unfavourable comments computer scientists reserve for Fortran, that language – certainly in its most recent 90/95 version – is eminently suitable for numerical analysis. This is partly because of the large volume of existing Fortran scientific subroutines (and intrinsic functions), although most of these are now also available for C++. C++ appears to be more powerful but also harder to learn, and the code is not as readily understandable as that for Fortran 90/95. In fact most of the past criticism of Fortran is now unfounded. Thus, one can have structures (collection of different data types under one name), pointers (enabling linked lists, for example) and recursion, a powerful tool for some applications. Another very useful feature of Fortran 90/95 is whole-array operations and array sections, which make programs more compact (eliminating many loops) and thus more easily readable. One might say that Fortran is more for the occasional programmer, while C++ is more suitable for writing general programs such as packages to solve a variety of simulation problems (see Chap. 16). Other languages can of course be used, such as Pascal or even Java, but these are rarities in digital simulation these days.

If Fortran is used, one strong recommendation is the use only of standard Fortran as far as practicable. Then, programs will be transportable. The example programs in the Appendices should all run on any computer with a standard Fortran 90/95 compiler. There is always the temptation to use extra features provided by one's local compiler. This entails the need to change

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 273–275 (2005)
www.springerlink.com

a number of programs when moving to another installation (or changing computers). Actually, even the example codes in the Appendices are not strictly standard. Adhering to the letter of the standard is not easy, as one finds when using the language F [392]. This is a standard-only compiler, and it makes programming difficult, also because it is not downward compatible, that is, it allows only the new features defined under Fortran 90/95. For example, all subroutines used in a given programming unit must be described in an interface. This is very tedious, and no other Fortran 90/95 compiler known to the present writer demands this. A useful text, one of many on Fortran 90/95, is that by Metcalf and Reid [393].

15.2 Debugging

It is well known that more time is spent correcting programs than actually planning or typing them in. With very large system programs, it is assumed [268] that some errors remain, and this is probably true for large simulation packages. However, the relatively short programs written for single applications by an electrochemist should be free of errors, and they are mostly rather easy to find.

Syntax errors are flagged by the compiler and are quickly eliminated. When a program is syntactically correct and compiled (and linked) successfully, it may still contain errors and these are harder to find. The use of `implicit none` is strongly recommended, as it allows the detection of typing errors in variable names. Clearly structured, modular programming also helps avoid errors or helps to localise them when they occur.

All variables must be given a value before being used. Some implementations start a program with all variables set to zero. It is tempting but dangerous to use this, as other installations leave previous values in all memory locations. Also it is good practice to set the compiler to checking array bounds. Some compilers allow a program to go beyond array bounds, and this can cause strange errors that are hard to trace. Checking slows down the program execution, but the check can be switched off when the program is finally considered correct. This can be called error prevention. Programming style can also help. There should be just sufficient comments to explain the less obvious lines, without causing distraction. The use of array sections, whenever whole arrays are addressed, is strongly recommended. For example, copying one array into another can be done simply by the statement

```
D = C
```

if they have the same declared dimensions, but it is better to explicitly indicate the index ranges with

```
D(1:N) = C(1:N)
```

This may be a little tedious but prevents some errors.

Finding program bugs is an art but certain techniques will help. The simplest method, after unsuccessfully reading through the problem program, is to explain the program to someone else. Often, one sees the error while doing this. This is called "egoless programming" [567]. Then there are diagnostic tools. Most Fortran implementations have a debugging facility which, when enabled, allows running a program with stops at strategic places, at which one can display some variable values. These tools can be a little unwieldy. A simpler method is to insert `print` statements at suspect places, narrowing them down until the error has been cornered. These days, extensive output is no problem, since we work at screen terminals and thus do not have to handle large volumes of paper. One very useful method is to display concentration values. One need not print them all, a select number usually being enough. For example, concentrations at all the edges of a domain are of interest. Often, one sees obviously incorrect values, which then point to the part of the program containing the error.

A difficult situation is a new simulation, with unknown results. How can we be sure that the results are correct? Often, the simulated system has special cases with known results; these should of course be checked. If we are developing a new simulation method, it can be checked against others known to work. For critical work, it may be necessary to write several different programs – perhaps written by different people – and to make sure that all converge to the same results as simulation intervals approach zero. A new program should be treated with suspicion, as if it were certain to contain bugs, even (or especially) if the results look "good". It is often possible to reduce the input parameters such that the results are known.

15.3 Libraries

One finds that a number of subroutines are used repeatedly in different programs and these are best placed into a library, possibly already compiled (object library), or in the form of a text library. There is a number of such routines discussed in Appendix C. The code `stuff.f90` is a module that the present author uses with every program by naming it in the compilation. As is seen, it defines some precision types, gives π to two different precision levels, etc. This makes the types in all programs and subroutines compatible with each other. The functions and subroutines are all in a large precompiled object library. These routines are all thoroughly tested.

16 Simulation Packages

Not every electrochemist wishes to write his or her own simulation programs, and there are a number of ready-made programs that can be obtained through the Internet or otherwise, some commercial, some free, and some that are online programs. These can be convenient but all have some limitations of various kinds. There have been several reviews describing these packages [104, 114, 523].

The following contains a number of Internet addresses. These are all active at the time of writing (September 2004) but some of them may not remain so indefinitely.

If using the MOL approach (Chap. 9), one needs to solve a set of *ode*s, if the boundary conditions can be incorporated into the differential equations. This will be the case with some simple boundary conditions such as those for the potential jump experiment. There are a number of freely available *ode* solvers, among them `VODE` [3], `LSODE` [4] and `PSODE` [553]. Some of these have been compared for efficiency [5,196], where also the sparse solvers `MA28` [1, 215] and `Y12` [584] were compared. Many of these routines can be downloaded from the `netlib` site [2]. If the boundary conditions (as is usual) are discretised in the form of algebraic equations, then they form, together with the system of *ode*s, a system of DAEs, as also mentioned in Chap. 9. For these, there is the program `DASSL`, described first by Petzold in 1983 [441], and again in the text of Brenan et al. [130], and can be downloaded from [6]. It uses BDF to solve the system.

There are some non-electrochemical *pde* solvers that some find convenient, such as `PHREEQC` [7] or the elliptic equation solver (useful for potential field or steady state computations) `PLTMG` [8], using the multigrid algorithm [40, 45, 391, 526] for increased speed. Some adherents of FEM use `FEMLAB` [9], while some French workers [87, 238, 239, 240, 241, 322, 491] prefer `Flux expert` [10]. The Spanish Horno group (see Chap. 9, Sect. 9.9 for many references to this) casts the electrochemistry into an electrical model and uses `PSPICE` [11] to solve the resulting network model. The same method is used by a Chinese group [205, 208, 209, 210], who in fact have written a general purpose electrochemical simulator around this technique (see below). Possibly the earliest use of a general simulator was that by Klinger et al. [339], who used the simulator `S/360 CSMP` of IBM, written for the IBM/360 machine.

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 277–279 (2005)
www.springerlink.com

CSMP was meant to simulate control processes. The authors used it to simulate CV of an adsorbed species.

Of greatest interest here is the group of programs that to lesser or greater degree solve a variety of electrochemical simulation problems. There are quite a number of these, a few of them somewhat prominent and commercial, and some more or less private but accessible to others.

ELSIM [90, 92, 99] is freely available from the author [12]. It has been updated since its earliest version around 1992, but is still DOS-based, that is, there is not a Windows version as yet. It accepts input in the form of reaction equations, in which case the program itself generates the governing equations; or the user can enter the governing equations directly. ELSIM is not limited to a discrete number of mechanisms or experiments, these being determined by what the user enters. Even the method used for simulation can be chosen (within some limits, indicated by the program when necessary). It is written in C++ and the source code is available and can be modified by the user.

DigiSim [482], sold by Bioanalytical Systems, Inc. [13], is also a general simulator, although it only offers CV simulations. It is general in the sense that any mechanism can be entered by the user and the program does the rest. It can be "tricked" into a few other simulation cases such as a potential step (by making the sweep very fast and applying a long holding time after the sweep) or even an electrochemical luminescence experiment by setting a huge diffusion coefficient for a fictitious species that is actually the light source [336]. However, CV is DigiSim's forte. The program is Windows-based and easy to use, but can be led into accepting nonsensical input, as reported in a review [137]. This has possibly been rectified since then. The program is written in C++.

One of the authors of DigiSim has now produced his own separate program, DigiElch, available through a web site [14]. Presumably it functions much like DigiSim.

Another well known simulator is attached to the text book by Gosser [274] on a diskette. It has also been reviewed [136, 522]. The program source code is written in Pascal. The program is now also available from a web page [15], where it is in its newer, patched, form. As the names of its two programs (`CVSIM` and `CVPLOT`) imply, it simulates only CV experiments.

The program `Polar`, apparently now called `Polarograph`, has been vigorously promoted by its author and can be downloaded as a free stripped-down version (and ordered in the complete form) [16]. The program has been reviewed in [17].

Speiser et al. began in 1989 to write the general simulation code EASI (for ElectroAnalytical SImulation), also extended to EASIEST, as parameter estimation was added. These are described in a series of papers [517, 518, 519, 520], and the code, which was in C++ and meant to run under the Unix operating system, was available from the author. The project has now been

terminated, in favour of a new, more general one, `Echem++` [372,373], which is still under development but promises to become a very general "problem solving environment" for electrochemical simulation. It is also described at a web site [18].

There are two online simulators to the author's knowledge, that of Alden et al. [19] and of Ohta [20], which can be used by anyone for a set of simulation situations.

Apart from the above, there is a plethora of more or less publically accessible electrochemical simulators. Carlo Nervi continues with his `ESP` package [21]. Elsyca presents the program `PIRODE` for industrially oriented electrochemical simulations [22], also described in a series of articles [192,551,552]. The same team produces the package `MIoTraS`, referred to in [544]. Sheehan et al. make a CV simulator tutorial available [23]. Simtel offers `VirtualCV` [24], and EE&G Princeton Applied offers `CONDESIM` [25], simulating several experiment types. Other packages are `TRANSIENT` [26], `SIMULA`, described in [492] as part of the "Seraphim project" but without any details being given. A Chinese group developed a package called `EEGNA` (exponentially expanded grid network approach) [205, 208, 209, 210]. The local `ECL-PACKAGE` (not accessible to others) is used by Svir and coworkers [542] for simulations of electrochemical luminescence experiments. Similar local general programs are presented by Penar et al. [440] for rotating electrode simulations and finally, there is an OC package for LSV by Villa et al. [561]. Both the two last programs are available from the authors of these articles.

A Tables and Formulae

A.1 First Derivative Approximations

In the Table A.1, the coefficients are multiplied by m (given in the first coefficients column), so as to get whole numbers. Only those approximations that are likely to be used are included.

Table A.1. $m\beta$ (3.14) for multi-point first derivatives. The notation $y_i'(n)$ means the approximation at point i using n points numbered $1 \ldots n$

	m	y_1	y_2	y_3	y_4	y_5	y_6	y_7	Order
$y_1'(2)$	1	−1	1						h
$y_2'(2)$	1	−1	1						h
$y_1'(3)$	2	−3	4	−1					h^2
$y_2'(3)$	2	−1	0	1					h^2
$y_3'(3)$	2	1	−4	3					h^2
$y_1'(4)$	6	−11	18	−9	2				h^3
$y_2'(4)$	6	−2	−3	6	−1				h^3
$y_3'(4)$	6	1	−6	3	2				h^3
$y_4'(4)$	6	−2	9	−18	11				h^3
$y_1'(5)$	12	−25	48	−36	16	−3			h^4
$y_2'(5)$	12	−3	−10	18	−6	1			h^4
$y_3'(5)$	12	1	−8	0	8	−1			h^4
$y_4'(5)$	12	−1	6	−18	10	3			h^4
$y_5'(5)$	12	3	−16	36	−48	25			h^4
$y_1'(6)$	60	−137	300	−300	200	−75	12		h^5
$y_2'(6)$	60	−12	−65	120	−60	20	−3		h^5
$y_3'(6)$	60	3	−30	−20	60	−15	2		h^5
$y_4'(6)$	60	−2	15	−60	20	30	−3		h^5
$y_5'(6)$	60	3	−20	60	−120	65	12		h^5
$y_6'(6)$	60	−12	75	−200	300	−300	137		h^5
$y_1'(7)$	60	−147	360	−450	400	−225	72	−10	h^6
$y_7'(7)$	60	10	−72	225	−400	450	−360	147	h^6

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 281–287 (2005)
www.springerlink.com

A.2 Current Approximations

These are simply obtained by using the appropriate coefficients in the above table, for the $y_1'(n)$ form chosen. For example, the two-point form is (3.23)

$$G \approx \frac{1}{H}\left(-C_0 + C_1\right) \tag{A.1}$$

while the three-point formula is (3.24),

$$G \approx \frac{1}{2H}\left(-3C_0 + 4C_1 - C_2\right) \ . \tag{A.2}$$

Note that all these have been cast as the function $\mathcal{G}(C, n, H)$, defined in Chap. 3, page 39.

A.3 Second Derivative Approximations

These are shown In Table A.2. As in Table A.1, the coefficients are multiplied by m (given in the first coefficients column), so as to get whole numbers.

Table A.2. Coefficients for some chosen multi-point second derivatives

	m	y_1	y_2	y_3	y_4	y_5	y_6	Order
$y_2''(3)$	1	1	−2	1				h^2
$y_3''(5)$	12	−1	16	−30	16	−1		h^4
$y_2''(6)$	12	10	−15	−4	14	−6	1	h^4
$y_5''(6)$	12	1	−6	14	−4	−15	10	h^4

A.4 Unequal Intervals

Here, the algorithm described in Chap. 3, Sect. 3.8, implemented in the very general subroutine `U_DERIV` referred to in Appendix C can provide both the derivatives on an arbitrarily spaced set of points (x, u). However, the reader may wish to restrict the expressions to those involving only up to four points (for which there are some good arguments, see Chap. 8, Sect. 8.4). This can be coupled with current approximations using up to four points. For this number of points, the expressions are not unreasonably long, and a few useful ones are therefore presented here.

As in Chap. 3, the notation used is that of a number n of positions $x_1 \ldots x_n$, at which some function values, respectively $u_1 \ldots u_n$ are defined,

and the derivative refers to point i out of the n. We also have a set of displacements $h_k = x_k - x_i$. In each case the zero displacement h_i is missing from the set. The following formulae (or some of them) have been given previously by Gavaghan [260] and Rudolph [478] but with different notation and different convention for the displacements.

A.4.1 First Derivatives

We have four current approximations $u_1'(n)$. Each one is given as a linear sum,

$$u_1'(n) \approx \sum_{k=1}^{n} \beta_k u_k \tag{A.3}$$

– $n = 2$ ($u_1'(2)$) is the same as for equal intervals, first-order

$$\begin{aligned} \beta_1 &= -1/h_2 \\ \beta_2 &= 1/h_2 \end{aligned} \tag{A.4}$$

– $n = 3$ ($u_1'(3)$), second-order

$$\Delta = \tfrac{1}{2}\left(h_2 h_3 (h_3 - h_2)\right) \tag{A.5}$$

and

$$\begin{aligned} \beta_2 &= h_3^2/2\Delta \\ \beta_3 &= -h_2^2/2\Delta \\ \beta_1 &= -(\beta_2 + \beta_3) \end{aligned} \tag{A.6}$$

– $n = 4$ ($u_1'(4)$), third-order

$$\Delta = \frac{1}{12}\Big(-h_2h_3^2h_4^3 + h_2h_3^3h_4^2 + h_3h_2^2h_4^3 - h_3h_2^3h_4^2 - h_4h_2^2h_3^3 + h_4h_2^3h_3^2\Big) \tag{A.7}$$

and

$$\begin{aligned} \beta_2 &= \frac{-h_3^2h_4^3 + h_3^3h_4^2}{12\Delta} \\ \beta_3 &= \frac{h_2^2h_4^3 - h_2^3h_4^2}{12\Delta} \\ \beta_4 &= \frac{-h_2^2h_3^3 + h_2^3h_3^2}{12\Delta} \\ \beta_1 &= -(\beta_2 + \beta_3 + \beta_4)\,. \end{aligned} \tag{A.8}$$

A.4.2 Second Derivatives

Second derivatives are represented in the linear form

$$u_i''(n) \approx \sum_{k=1}^{n} \alpha_k u_k \tag{A.9}$$

– $n = 3$ ($u_1''(3)$), first-order one-sided formula

$$\Delta = \tfrac{1}{2}\,(h_2 h_3 (h_3 - h_2)) \tag{A.10}$$

and

$$\begin{aligned} \alpha_2 &= -h_3/\Delta \\ \alpha_3 &= h_2/\Delta \\ \alpha_1 &= -(\alpha_2 + \alpha_3) \end{aligned} \tag{A.11}$$

– $n = 3$ ($u_2''(3)$), first-order central formula

$$\Delta = \tfrac{1}{2}\,(h_1 h_3 (h_3 - h_1)) \tag{A.12}$$

and

$$\begin{aligned} \alpha_1 &= -h_3/\Delta \\ \alpha_3 &= h_1/\Delta \\ \alpha_2 &= -(\alpha_1 + \alpha_3) \end{aligned} \tag{A.13}$$

– $n = 4$ ($u_2''(4)$), second-order

$$\Delta = \frac{1}{12}\Big(-h_1 h_3^2 h_4^3 + h_1 h_3^3 h_4^2 + h_3 h_1^2 h_4^3 - h_3 h_1^3 h_4^2 - h_4 h_1^2 h_3^3 + h_4 h_1^3 h_3^2\Big) \tag{A.14}$$

and

$$\begin{aligned} \alpha_1 &= \frac{h_3 h_4^3 - h_3^3 h_4}{6\Delta} \\ \alpha_3 &= \frac{-h_1 h_4^3 + h_1^3 h_4}{6\Delta} \\ \alpha_4 &= \frac{h_1 h_3^3 - h_1^3 h_3}{6\Delta} \\ \alpha_2 &= -(\alpha_1 + \alpha_3 + \alpha_4)\,. \end{aligned} \tag{A.15}$$

A.5 Jacobi Roots for Orthogonal Collocation

The table below provides the roots of the Jacobi polynomials used as node points in orthogonal collocation, for some values of N. Values for $X = 0$ ($i = 0$) and $X = 1$ ($i = N + 1$) (the values are 0 and 1, resp.) are not included. The roots were computed using the subroutine `JCOBI`, modified from the original of Villadsen and Michelsen [562], discussed in Appendix C, using for a given N the call
`CALL JCOBI (N+1, N, 0, 0, 0.0_dbl, 0.0_dbl,...)`.
Roots for higher N, if required, can be computed using this subroutine.

Table A.3. Jacobi polynomial roots, $N = 3 \ldots 6$

$N = 3$	$N = 4$	$N = 5$	$N = 6$
0.11270166537926	0.06943184420297	0.04691007703067	0.03376524289842
0.50000000000000	0.33000947820757	0.23076534494716	0.16939530676687
0.88729833462074	0.66999052179243	0.50000000000000	0.38069040695840
	0.93056815579703	0.76923465505284	0.61930959304160
		0.95308992296933	0.83060469323313
			0.96623475710158

Table A.4. Jacobi polynomial roots, $N = 7 \ldots 10$

$N = 7$	$N = 8$	$N = 9$	$N = 10$
0.02544604382862	0.01985507175123	0.01591988024619	0.01304673574141
0.12923440720030	0.10166676129319	0.08198444633668	0.06746831665551
0.29707742431130	0.23723379504184	0.19331428364970	0.16029521585049
0.50000000000000	0.40828267875218	0.33787328829810	0.28330230293538
0.70292257568870	0.59171732124782	0.50000000000000	0.42556283050918
0.87076559279970	0.76276620495816	0.66212671170190	0.57443716949082
0.97455395617138	0.89833323870681	0.80668571635030	0.71669769706462
	0.98014492824877	0.91801555366332	0.83970478414951
		0.98408011975381	0.93253168334449
			0.98695326425859

A.6 Rosenbrock Constants

The Rosenbrock method is described for *ode*s in Chap. 4 and for electrochemical simulations, that is, DAEs, in Chap. 9. There are four variants, two of

Table A.5. Jacobi polynomial roots, $N = 11 \ldots 14$

$N = 11$	$N = 12$	$N = 13$	$N = 14$
0.01088567092697	0.00921968287664	0.00790847264071	0.00685809565159
0.05646870011595	0.04794137181476	0.04120080038851	0.03578255816821
0.13492399721298	0.11504866290285	0.09921095463335	0.08639934246512
0.24045193539659	0.20634102285669	0.17882533027983	0.15635354759416
0.36522842202383	0.31608425050091	0.27575362448178	0.24237568182092
0.50000000000000	0.43738329574427	0.38477084202243	0.34044381553606
0.63477157797617	0.56261670425573	0.50000000000000	0.44597252564633
0.75954806460341	0.68391574949909	0.61522915797757	0.55402747435367
0.86507600278702	0.79365897714331	0.72424637551822	0.65955618446394
0.94353129988405	0.88495133709715	0.82117466972017	0.75762431817908
0.98911432907303	0.95205862818524	0.90078904536665	0.84364645240584
	0.99078031712336	0.95879919961149	0.91360065753488
		0.99209152735929	0.96421744183179
			0.99314190434841

them second order with respect to the time interval, and two of them third-order, that are considered in these chapters. Although it is clear that only two variants recommend themselves, the constants for all four are given her. For the notation and the meaning of the variant names, see these chapters. The notation, in some cases, is not that of the (cited) sources. Omitted constants can be taken as zero.

RO2, [474], see Sect. 4.12
$\gamma = 1 - \frac{1}{2}\sqrt{2}$
$a_{21} = \frac{1}{2}(\sqrt{2} - 1); \quad \alpha_2 = a_{21}$
$m_1 = 0; \quad m_2 = 1$

ROS2, [347]
$\gamma = 1.707106781186547$
$\gamma_1 = \gamma; \quad \gamma_2 = -\gamma$
$\alpha_1 = 0; \quad \alpha_2 = 1$
$a_{21} = 0.5857864376269050$
$c_{21} = -1.171572875253810$
$m_1 = 0.8786796564403575; \quad m_2 = 0.2928932188134525$

ROWDA3, [473,100]
$\gamma = 0.4358665215084590$
$\gamma_1 = \gamma$
$\gamma_2 = 0.6044552840655590$
$\gamma_3 = 6.379788799344883$
$\alpha_2 = 0.7; \quad \alpha_3 = 0.7$
$a_{21} = 1.605996252195329$

$a_{31} = a_{21}; \quad a_{32} = 0$
$c_{21} = 0.8874044410657833$
$c_{31} = 23.98747971635036$
$c_{32} = 5.263722371562129$
$m_1 = 2.236727045296590$
$m_2 = 2.250067730969644$
$m_3 = -0.2092514044390320$

ROS3P, [347, 348]
$\gamma = 0.7886751345948129$
$\gamma_1 = \gamma$
$\gamma_2 = -0.2113248654051871$
$\gamma_3 = -1.077350269189626$
$a_{21} = 1.267949192431123; \quad a_{31} = a_{21}$
$c_{21} = -1.607695154586736$
$c_{31} = -3.464101615137755$
$c_{32} = -1.732050807568877$
$m_1 = 2$
$m_2 = 0.5773502691896258$
$m_3 = 0.4226497308103742$

B Some Mathematical Proofs

B.1 Consistency of the Sequential Method

As described in Chap. 5, for the simulation of a first order homogeneous chemical reaction (*hcr*) coupled to diffusion such as the Reinert-Berg mechanism (5.11) we have the governing equation

$$\frac{\partial C}{\partial T} = \frac{\partial^2 C}{\partial X^2} - KC \tag{B.1}$$

for the explicit point method, and it discretises to

$$C_i' = C_i + \lambda \left(C_{i-1} - 2C_i + C_{i+1}\right) - K\delta T C_i \tag{B.2}$$

containing, on the right-hand side, a term for diffusion and one for the *hcr*. The sequential method first calculates intermediate new concentrations, only affected by diffusion,

$$C_i^* = C_i + \lambda \left(C_{i-1} - 2C_i + C_{i+1}\right) \tag{B.3}$$

and then allows the *hcr* to act on this

$$C_i' = C_i^* - K\delta T C_i^* \tag{B.4}$$

giving, after combining the two,

$$C_i' = (1 - K\delta T)\left\{C_i + \lambda \left(C_{i-1} - 2C_i + C_{i+1}\right)\right\} \ . \tag{B.5}$$

The proof of the consistency of this procedure [485] goes as follows. Multiply (B.1) by e^{KT}:

$$\frac{\partial C}{\partial T}\, e^{KT} = e^{KT} \frac{\partial^2 C}{\partial X^2} - KCe^{KT} \ . \tag{B.6}$$

Now, since

$$\frac{\partial}{\partial T}\left(Ce^{KT}\right) = \frac{\partial C}{\partial T}\, e^{KT} + KCe^{KT} \ , \tag{B.7}$$

(B.6) can be written as

$$\frac{\partial}{\partial T}\left(Ce^{KT}\right) = e^{KT} \frac{\partial^2 C}{\partial X^2} \ . \tag{B.8}$$

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 289–297 (2005)
www.springerlink.com

The left-hand side can be approximated as the forward difference (at point i along X)

$$\frac{\partial}{\partial T}\left(Ce^{KT}\right) \approx \frac{C_i' e^{K(T+\delta T)} - Ce^{KT}}{\delta T} \tag{B.9}$$

(recalling that C_i is $C_i(T)$ and that $C_i' = C_i(T + \delta T)$) and combining this with (B.8) and discretising the second derivative as usual (5.2, page 74), this leads to

$$C_i' e^{K(T+\delta T)} = C_i e^{KT} + \lambda e^{KT}\left(C_{i-1} - 2C_i + C_{i+1}\right) \tag{B.10}$$

which becomes, upon dividing throughout by $e^{K(T+\delta T)}$ and rearranging,

$$C_i' = e^{-K\delta T}\left\{C_i + \lambda\left(C_{i-1} - 2C_i + C_{i+1}\right)\right\} . \tag{B.11}$$

This converges to (B.5) as $K\delta T \to 0$. The sequential method is therefore mathematically consistent and this is the reason that it works rather well, within the limits of the approximation $e^{-K\delta T} \approx 1 - K\delta T$.

B.2 The Feldberg Start for BDF

The simple start for BDF, as adopted by Mocak and Feldberg [402] coupled with the subsequent subtraction of half a time interval, as described in Chap. 4, Sect. 4.8.1, was found [142, 155] to be mathematically consistent. This is proved as follows.

The proof is given for a general *ode*. It also applies to a system of *ode*s and thus to the system of equations resulting from the discretisation of a *pde*. Let the equation to be solved be

$$u' = f(u), \quad u(0) = u_0 \tag{B.12}$$

with $f(u)$ being some unspecified function. This is solved using BDF (see Chap. 4). The contention is that after a number n time steps each of length h the time, which should be equal to nh, has in fact been shifted by a fraction s of the time step length h, that is, by $s \times h$, and that this converges, after a number of steps, to $-0.5h$, justifying the Feldbergian correction.

The numerical solution yields a sequence of approximations to $u(t)$ at the times $t = h, 2h, \ldots$ denoted respectively as $u_1, u_2, \ldots$. For convenience, we write f_n to denote $f(u_n)$.

In what follows here, in order to be consistent with [143] (and normal computer science usage), BDF is described on a number of levels, rather than the number of points in time. Thus, the symbol k now refers to levels, and is less by 1 than the k used in other parts of this book. So, three-point BDF corresponds to $k = 2$, etc.

First, the expression to be solved is developed, for a simple case. We seek a solution to the *ode* (B.12), using 2-level BDF. The BDF expression at the nth step is

$$\frac{u_{n-2} - 4u_{n-1} + 3u_n}{2h} = f(u_n) \tag{B.13}$$

or

$$u_n = -\tfrac{1}{3}u_{n-2} + \tfrac{4}{3}u_{n-1} + \tfrac{2}{3}hf(u_n)\ . \tag{B.14}$$

For the very first step, $n = 1$, u_{-1} is lacking, and with the simple start (see Chap. 4, Sect. 4.8.1) one simply substitutes u_0. The result is then

$$u_1 = u_0 + \tfrac{2}{3}hf_1 \tag{B.15}$$

(writing f_1 instead of $f(u_1)$). This is clearly equivalent to a BI step of length $\frac{2}{3}h$, and thus, one-third of an interval has been "lost", and $s_1 = -\frac{1}{3}$.

At step 2, using (B.13) and substituting for u_1 as from (B.15), the equation is

$$u_2 = u_0 + \tfrac{8}{9}hf_1 + \tfrac{2}{3}hf_2 \tag{B.16}$$

and we now have a total advance of $\frac{14}{9}h$, or shift $s_2 = -\frac{4}{9}$.

At step $n-1$, let the expression be

$$u_{n-1} = u_0 + p_1hf_1 + p_2hf_2 + \cdots + p_{n-1}hf_{n-1} \tag{B.17}$$

and denote the total advance (given in units of h) in t as

$$a_{n-1} = \sum_{i=1}^{n-1} p_i\ . \tag{B.18}$$

At the next step, we have a new series

$$u_n = u_0 + q_1hf_1 + q_2hf_2 + \cdots + q_nhf_n \tag{B.19}$$

(with $q_n = 2/3$) and a new sum of advances

$$a_n = \sum_{i=1}^{n} q_i\ . \tag{B.20}$$

From (B.13), we note that generation of u_n is a linear combination of the p and q sequences, with the added last term, $\frac{2}{3}hf_n$. Thus, we have the recursive expression for the advances,

$$a_n = -\tfrac{1}{3}a_{n-2} + \tfrac{4}{3}a_{n-1} + \tfrac{2}{3}\ . \tag{B.21}$$

There is no advance before the first step, so that

$$a_i = 0, \quad i \le 0\ . \tag{B.22}$$

By definition,

$$s_n = a_n - n \tag{B.23}$$

and this makes, given (B.22)

$$s_i = -i, \quad i \leq 0\,. \tag{B.24}$$

Substituting (B.23) into (B.21) finally yields the recursive expression for all s_n,

$$3s_n - 4s_{n-1} + s_{n-2} = 0 \tag{B.25}$$

with starting values for s_n obtained from (B.24). The above treatment can be extended to higher-level forms, see below.

Up to this point, this has been presented in [142]. In that paper, this was then followed by computer calculations showing that for $k = 2, 3, 4$, the s_n values converge to -0.5. There is, however, a mathematical proof [155].

The first few further recursive equations to be solved are

$$k{=}3: \quad 11s_n - 18s_{n-1} + 9s_{n-2} - 2s_{n-3} = 0 \tag{B.26}$$

$$k{=}4: \quad 25s_n - 48s_{n-1} + 36s_{n-2} - 16s_{n-3} + 3s_{n-4} = 0 \tag{B.27}$$

$$k{=}5: \quad 137s_n - 300s_{n-1} + 300s_{n-2} - 200s_{n-3} + 75s_{n-4} - 12s_{n-5} = 0 \tag{B.28}$$

$$k{=}6: \quad 147s_n - 360s_{n-1} + 450s_{n-2} - 400s_{n-3} + 225s_{n-4} - 72s_{n-5} + 10s_{n-6} = 0 \tag{B.29}$$

covering all the stable cases. The general form is

$$\alpha_0 s_n + \alpha_1 s_{n-1} + \alpha_2 s_{n-2} + \cdots + \alpha_k s_{n-k} = 0, \tag{B.30}$$

starting with values s_0, s_{-1}, etc., as given above in (B.24). Consider the generating function

$$\begin{aligned}
\alpha_0 \cdot \sum_{n=0}^{\infty} s_n t^n &= \sum_{n=0}^{k-1} \alpha_0 s_n t^n + \sum_{n=k}^{\infty} \alpha_0 s_n t^n \\
&= \sum_{n=0}^{k-1} \alpha_0 s_n t^n - \sum_{n=k}^{\infty} \sum_{i=1}^{k} \alpha_i s_{n-i} t^n \\
&= \sum_{n=0}^{k-1} \alpha_0 s_n t^n - \sum_{i=1}^{k} \alpha_i t^i \sum_{n=k-i}^{\infty} s_n t^n \\
&= \sum_{n=0}^{k-1} \alpha_0 s_n t^n - \sum_{i=1}^{k} \alpha_i t^i \sum_{n=0}^{\infty} s_n t^n + \sum_{i=1}^{k-1} \alpha_i t^i \sum_{n=0}^{k-i-1} s_n t^n \\
&= \sum_{i=0}^{k-1} \alpha_i t^i \sum_{n=0}^{k-i-1} s_n t^n - \sum_{i=1}^{k} \alpha_i t^i \sum_{n=0}^{\infty} s_n t^n \quad .
\end{aligned} \tag{B.31}$$

Thus

$$\sum_{n=0}^{\infty} s_n t^n = \frac{\displaystyle\sum_{i=0}^{k-1} \alpha_i t^i \sum_{n=0}^{k-i-1} s_n t^n}{\displaystyle\sum_{i=0}^{k} \alpha_i t^i} \quad . \tag{B.32}$$

We now want to find a single power series in t to replace the expression on the right-hand side of (B.32), and its terms will then be equivalent with those of the left-hand side and yield the desired s_n. This is done by splitting (B.32) into partial fractions. An example follows here, namely the case $k = 2$ from (B.25) above. We now solve the recursive equations for s_{n-1}, for all $n \geq 2$. We have

$$\sum_{n=0}^{\infty} s_{n-1} t^n = \frac{\displaystyle\sum_{i=0}^{1} \alpha_i t^i \sum_{n=0}^{1-i} s_{n-1} t^n}{\displaystyle\sum_{i=0}^{2} \alpha_i t^i} = \frac{3s_{-1} - 4s_{-1}t + 3s_0 t}{3 - 4t + t^2} \,, \tag{B.33}$$

i.e.,

$$\sum_{n=0}^{\infty} s_{n-1} t^n = \frac{1}{2} \frac{-s_{-1} + 3s_0}{1 - t} + \frac{9}{2} \frac{s_{-1} - s_0}{3 - t} \,. \tag{B.34}$$

Hence, expanding as a power series in t,

$$\sum_{n=0}^{\infty} s_{n-1} t^n = \frac{1}{2} \sum_{n=0}^{\infty} \big(-s_{-1} + 3s_0 + 3(s_{-1} - s_0) \frac{1}{3^n}\big) t^n \tag{B.35}$$

and s_{n-1} is obtained by identifying the coefficients of the two sums. The solution to (B.25) is then

$$s_n = \frac{1}{2} \left(-s_{-1} + 3s_0\right) + \frac{1}{2} \left(s_{-1} - s_0\right) \frac{1}{3^n} \,. \tag{B.36}$$

With increasing n, the second term on the right-hand side vanishes and together with (B.23) above (that is, $s_{-1} = 1$ and $s_0 = 0$); s_n converges to $-1/2$. The same procedure applied to the higher k values yields the following solutions:

$$k{=}3: \quad s_n = \frac{1}{6} \left(2s_{-2} - 7s_{-1} + 11s_0\right) + O(2.35^{-n}) \tag{B.37}$$

$$k{=}4: \quad s_n = \frac{1}{12} \left(-3s_{-3} + 13s_{-2} - 23s_{-1} + 25s_0\right) + O(1.78^{-n}) \tag{B.38}$$

$$k{=}5: \quad s_n = \frac{1}{60} \left(12s_{-4} - 63s_{-3} + 137s_{-2} - 163s_{-1} + 137s_0\right) + O(1.41^{-n}) \tag{B.39}$$

$$k{=}6: \quad s_n = \frac{1}{60} \left(-10s_{-5} + 62s_{-4} - 163s_{-3} + 237s_{-2} - 213s_{-1} + 147s_0\right) + O(1.16^{-n}) \quad . \tag{B.40}$$

The numbers 3, 2.35, 1.78, 1.41, 1.16, are the numerically smallest of the polynomial roots, all of which are shown in Table B.1, extending the range of k.

Table B.1. Roots of the polynomials for $k = 1 \ldots 8$

k							
1	1						
2	1	3					
3	1	2.35	2.35				
4	1	2.61	1.78	1.78			
5	1	2.39	2.39	1.41	1.41		
6	1	2.46	2.11	2.11	1.16	1.16	
7	1	2.35	2.35	1.85	1.85	0.978	0.978
8	1	2.37	2.18	2.18	1.64	1.64	0.845	0.845

Now, $s_i = -i$ for $i \leq 0$, so

$$\lim_{n\to\infty} s_n^{(k)} = -\frac{1}{2}\,. \tag{B.41}$$

for all five values of k, shown above, that is, $k \leq 6$.

The general solution to (B.30) is of the form

$$s_n = \sum_{i=1}^{k} P_i(n)\lambda_i^{-n} \tag{B.42}$$

where the λ_i coefficients are the complex roots of the polynomial

$$\alpha_0 + \alpha_1 t + \alpha_2 t^2 + \cdots + \alpha_k t^k = \alpha_0 \prod_{i=1}^{k} \left(1 - \frac{t}{\lambda_i}\right)\,. \tag{B.43}$$

See [525, Chap. 4] for a proof of this.

The convergence for $k \leq 6$ is due to the fact that the roots of the five polynomials

$$3 - 4t + t^2 \tag{B.44}$$

$$11 - 18t + 9t^2 - 2t^3 \tag{B.45}$$

$$25 - 48t + 36t^2 - 16t^3 + 3t^4 \tag{B.46}$$

$$137 - 300t + 300t^2 - 200t^3 + 75t^4 - 12t^5 \tag{B.47}$$

$$147 - 360t + 450t^2 - 400t^3 + 225t^4 - 72t^5 + 10t^6 \tag{B.48}$$

all are numerically equal to or greater than 1.

The coefficients in the above polynomials are those for the BDF forms, given in Appendix A, as the last entry in each group in Table A.1.

Note that in the cases $k > 6$ there appears at least one polynomial root which is numerically smaller than 1. The form of the general solution (B.42) therefore implies that the values s_n then do not converge to a finite value, as n tends towards infinity. This is in accord with the known instability of BDF for $k > 6$ [187].

Table B.2 shows a few shifts s_n for some k, and the convergence is clearly seen.

Table B.2. s_n values

n	$k = 2$	$k = 3$	$k = 4$
1	−0.333	−0.455	−0.520
2	−0.444	−0.562	−0.598
3	−0.481	−0.548	−0.520
4	−0.494	−0.519	−0.470
5	−0.498	−0.503	−0.474
6	−0.499	−0.499	−0.494
7	−0.500	−0.499	−0.504
8	−0.500	−0.499	−0.504
9	−0.500	−0.500	−0.501
10	−0.500	−0.500	−0.499

Convergence is slower for higher k, as is also implied by the values of the polynomial roots in Table B.1.

B.3 Similarity of the Feldberg Expansion and Transformation Functions

In Chap. 7, two ways of implementing unequal intervals were described. These were the Feldberg approach, in which exponentially expanding boxes are placed along the X-axis (7.16), and the transformation method (7.3). Here it will be shown that they are approximately equivalent, and the relation between their respective expansion parameters will be given.

The Feldberg expansion consists of a starting (first) box length of length H_1. Subsequent boxes are then defined such that box number i has length $\beta^{i-1} H_1$. See Fig. B.1 for this and the points spacing. Points spacing uses the transformation relations

$$Y = \ln(1 + aX) \tag{B.49}$$

and the reverse

$$X = (e^Y - 1)/a \, . \tag{B.50}$$

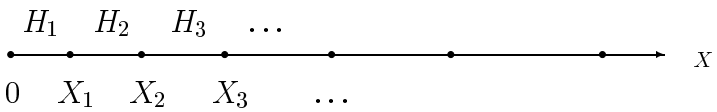


Fig. B.1. Points and box unequal spacing

We have at any point,

$$\begin{aligned} X_n &= H_1 + H_1\beta + H_1\beta^2 + \cdots + H_1\beta^{n-1} \\ &= H_1 \sum_{k=0}^{n-1} \beta^k \\ &= H_1 \sum_{k=0}^{n-1} \exp(k \ln \beta) \ . \end{aligned} \tag{B.51}$$

For large n, the sum approaches the integral and we have

$$X_n = H_1 \int_0^n \exp(k \ln \beta)\, dk \tag{B.52}$$

and this is readily integrated to yield

$$X_n = \frac{H_1}{\ln(\beta)} \left(\exp(n \ln \beta) - 1\right) \ . \tag{B.53}$$

This is of the form of (B.50), thus establishing the equivalence. Also, we can read off the relation between the parameters, being

$$a \equiv \ln \beta / H_1 \tag{B.54}$$

and also since $Y_n = n\delta Y$,

$$\delta Y \equiv \ln \beta \ . \tag{B.55}$$

It is noted that the derivation is an approximation, resting on the approximate equality of the sum and integral, and holding better for larger n. However, it might be useful, given the information of the values of, say, H_1 and β in a paper using the box method, to be able to translate it into the corresponding values of a and δY.

The second way of establishing the parameter relations focusses on three points (encompassing two boxes), X_{n-1}, X_n, X_{n+1} obtained from (B.50). If they also obey the box expansion formula, then a definite expression for β should be obtained from the ratio of the two box lengths,

$$\beta = \frac{X_{n+1} - X_n}{X_n - X_{n-1}} \tag{B.56}$$

which, substituting from (B.50), and reducing, becomes

$$\beta = \frac{e^{\delta Y} - 1}{1 - e^{-\delta Y}} \tag{B.57}$$

which reduces to

$$\beta = e^{\delta Y} \tag{B.58}$$

the same as (B.55). Then using

$$\delta Y = \ln(1 + aH_1) = \ln \beta \tag{B.59}$$

and the approximation $\ln(1 + aH_1) \approx aH_1$ holding for $aH_1 \ll 1$ (which will usually be the case), relation (B.54). If $aH_1 \ll 1$ does not hold, then the more correct relation will be

$$\beta = 1 + aH_1 \ . \tag{B.60}$$

C Procedure and Program Examples

Here some modules, procedures and whole programs are described, that may be useful to the reader, as they have been, to the author. They are all in Fortran 90/95 and start with a generally useful module, that will be used in most procedures and programs in the examples, and another module useful for programs using a Rosenbrock variant. The source texts (except for the two modules) are not reproduced here, but can be downloaded from the web site `www.springerlink.com/openurl.asp?genre=issue&issn=1616-6361&volume=666`
(the two lines form one contiguous URL!).

C.1 Example Modules

Module `STUFF`

In this module, the two `kind`s of `real` values, respectively single precision (6 decimals or a little better) and what used to be called double precision (here defined as 14 decimals or a little better), are given the names `sgl` and `dbl`, and the constant `pi` is made a parameter. As well, the (`real`) parameter `small` is set up. This is useful in those cases where something is tested for being close to zero.

```
module STUFF
! General-purpose module.
  implicit none
  integer, parameter         :: sgl=selected_real_kind(6),&
                                dbl=selected_real_kind(14)
  real(kind=dbl), parameter  :: small = 1.0E-08
  real(kind=dbl), parameter  :: pi = 3.14159265358979
end module STUFF
```

Module `ROSTUFF`

Rosenbrock methods require a set of constants, and as these are needed in several subroutines in a given program, it is convenient to gather them in a

Dieter Britz: *Digital Simulation in Electrochemistry*, Lect. Notes Phys. **666**, 299–311 (2005)
www.springerlink.com

module. Experiments have shown, that the two Rosenbrock variants, ROS2 and ROWDA3 (see Chap. 9), are about the best in the present context, responding without oscillations. Only these two have been included in the module. Readers wanting the constants for the two other methods RO2 and ROS3P, can find them in Appendix A, in the same unified notation as these two preferred variants.

```
module ROSTUFF
! Module for the Rosenbrock coefficients.
  use STUFF;   implicit none
  real(kind=dbl) :: gamma, gamma1, gamma2, gamma3, &
                    alpha1,  alpha2,  alpha3,      &
                    a21, a31, a32,                 &
                    c21, c31, c32, m1, m2, m3
CONTAINS
  subroutine ROCOEFFS (order)
  ! Sets the Rosenbrock coeffs for orders 2 or 3,
  ! Order 2 is ROS2 (Lang, 1995), order 3 is ROWDA3.
  use STUFF;  implicit none
  integer :: order

  gamma1  = 0;   gamma2 = 0;   gamma3 = 0 ! Zero defaults
  alpha1  = 0;   alpha2 = 0;   alpha3 = 0
  a21 = 0;   a31 = 0;   a32 = 0
  c21 = 0;   c31 = 0;   c32 = 0
  m1 = 0
  select case (order)
    case (2)
      gamma = 1.707106781186547_dbl
      gamma1 = 0;   gamma2 = - gamma
      alpha2 = 1
      a21 = 0.5857864376269050_dbl
      c21 = - 1.171572875253810_dbl
      m1  = 0.8786796564403575_dbl
      m2  = 0.2928932188134525_dbl
    case (3)
      gamma = 0.435866521508459_dbl
      gamma1 = gamma;  gamma2 = 0.6044552840655588_dbl
                       gamma3 = 6.3797887993448800_dbl
      alpha2 = 0.7_dbl;  alpha3 = 0.7_dbl
      a21 = 1.605996252195329_dbl
      a31 = a21;  a32 = 0
      c21 = 0.8874044410657823_dbl
      c31 = 23.98747971635035_dbl
      c32 = 5.263722371562130_dbl
```

```
      m1 = 2.236727045296589_dbl
      m2 = 2.250067730969645_dbl
      m3 = -0.209251404439032_dbl
    end select
  end subroutine ROCOEFFS
end module ROSTUFF
```

C.2 Procedures

Procedures are either functions or subroutines. A few, that recur in simulation programs, are presented here.

The Error Functions

In test programs, where the numerical solution is compared with the analytical solution, the latter often involves the error function erf or the complementary error function erfc. The latter could be obtained simply by subtracting erf from unity but a better approximation is obtained by the direct algorithm. The two routines, `ERF` and `ERFC`, were given to the author by a colleague, who probably obtained them from an IBM collection. They have been adapted to Fortran 90/95 by the author, and coupled to the above module. The comments in capitals are the original comments.

Current Approximations

The current value is obtained from the concentrations at any time as the dimensionless quantity G, the gradient dC/dX at $X = 0$ as an n-point forward difference (see Appendix A, or Sect. 3.4). This is conveniently computed by a function, here called `GOFUNC`. It, and the following function `COFUNC` use the function `GOBETA`, which supplies the n-point coefficients.

This function can be inverted to calculate C_0, given G; this is useful in the formulation of some boundary conditions. For example, to take a simple case, in chronopotentiometry, one has constant G and computes C_0 from that and the concentration profile. The function `COFUNC` does this job. Both G and H are needed, but they appear as the product, so that product is passed to the function.

Both these functions call on `GOBETA` for the coefficients. It is also useful in other contexts, such as the setting up of boundary value calculations for coupled systems.

For unequal intervals, see below, Sect. C.2.1.

`MAT_INV`

In some programs such as `CVRUCAT` or the subroutine `U_DERIV`, matrix inversion is needed. This is best done by using LU decomposition, as described in Press et al. (1986). The subroutine `MATINV` does this. It assumes a square matrix, of the exact size given, so the best way to call it is by using a section of that size, for example

```
call MATINV (mat(1:N,1:N), N)
```

As seen, if the matrix is only 2×2, it is inverted "manually". The two subroutines `DEC` and `SOL` are the usual LU decomposition routines, of which there are a number, freely available.

`MINMAX`

This subroutine is useful in linear sweep simulations, for calculating peak (or trough) current values and where they occur, from a trio of currents, in which the second is either larger, or smaller, than the other two. A parabola is fitted to the three points,

$$y = a_0 + a_1 x + a_2 x^2 \tag{C.1}$$

and the parabola's maximum or minimum is computed, as well as its position. The positions are assumed (by the subroutine) to be, respectively, at -1, 0 and $+1$, so that it is up to the calling program unit to scale the minmax position.

C.2.1 Procedures for Unequal Intervals

For arbitrarily spaced intervals, we require procedures for first and second derivatives, and some other subroutines.

`EE_FAC`

There is a need in some situations to generate a series of points, with the intervals between them exponentially expanding from a base value. This can be both in time or in space. It is often most convenient (see Chap. 7) to start with the base interval, H_1 (which is also the first value X_1 after zero), the last value, X_L, and the number of points N to be generated in that range. The expansion factor γ then makes

$$h_i = \gamma h_{i-1} \tag{C.2}$$

or, as in Chap. 7,

$$x_i = \frac{\gamma^i - 1}{\gamma - 1} \, . \tag{C.3}$$

The task is then to find the γ that fits the requirements. This can easily be done using a binary search, and the function `EE_FAC` is provided for this.

`SV_FAC`

There is another sequence of point positions of potential use, the S&V sequence, discussed in Chap. 7, page 108. This sequence is described by the recursion

$$h_i = h_{i-1}(1 + \alpha H_{i-1}/H_1) \; . \tag{C.4}$$

As with the exponentially expanding sequence, we start with a first interval H_1, decide on a furthest point X_L and choose the number of points in the sequence. The function `SV_FAC` computes the expansion factor, here called α. The task is then to find the α that fits the requirements.

`U_DERIV`

This subroutine is a general routine for computing the first or second derivative on a number n of points, referred to the ith one among that number, on an arbitrarily spaced grid of points. The derivatives are computed as a linear sum of terms, and the coefficients in that sum are also passed back, for use, for example, in the discretisation of boundary conditions or the spatial second derivative. The number of points is in principle unrestricted, but the routine will fail for values $n > 12$, where the accuracy abruptly drops. A value, in any case, exceeding about 8, is perhaps impractical. This routine can be used instead of the algebraic expressions shown in Chap. 7, or if n values greater than 4 are required.

The notation behind the routine is that a first or second derivative, respectively $u_i'(n)$ or $u_i''(n)$ is based on a number of function values $u_k, k = 1 \ldots n$ situated at positions $x_k, k = 1 \ldots n$, and computed as the sum $\sum_{k=1}^{n} \alpha_k u_k$, the α_k being the coefficients. The index i is the point, from 1 to n, to which the derivative is referred. The subroutine uses `MATINV`. Note that, as discussed in Chap. 3, page 46, the factorials in the denominator from the Taylor expansions are attached to the solution vector, and corrected for after inversion (in this case either by unity, for the first derivative, or $2! = 2$ for the second). This gives a slight increase in accuracy.

Current Approximations on Unequal Intervals

Once we have the subroutine `U_DERIV` shown above, it is simple to construct a more convenient function to calculate the current approximation, if that is all we want (that is, if we do not want the coefficients that make it up). The function `GU` does the job, calling the more complicated `U_DERIV` to do the hard work.

The "Inverse" Current Function

`U_DERIV` can be used to compute C_0, given the current (as in chronopotentiometry) and the concentration profile. As for equal intervals, the current approximation formula on n points is adapted, by the function `CU`.

Note that the use of both GU and CU is not restricted to unequal intervals; they can also be used with equal intervals, where we already have GOFUNC and COFUNC, given above. The present two functions will take a little more computing time, but this is normally a small part of any given simulation, where the recalculation of a concentration profile is most time consuming.

Current Integration on an Unequally Gridded Surface

In the case of the ultramicroelectrodes such as the disk electrode, it is necessary to integrate over the surface, and sometimes there will be unequally spaced points along the surface, as for example, in direct discretisation on an unequal grid in the example program UME_DIRECT. As mentioned in Chap. 12, it is found that due to the errors in the computed concentration values, the local fluxes are so inaccurate that any integration method better than the simple trapezium method is not justified. The routine U_TRAP is thus recommended here. It integrates local current densities, precalculated by using the above routine U_DERIV.

C.2.2 JCOBI

In Chap. 9, the method orthogonal collocation is described. It makes use of certain Jacobi polynomials, whose roots become the node points X, at which concentrations are defined. The subroutine JCOBI is an adaptation of the subroutine reproduced in the book by Villadsen and Michelsen [562], converting it to Fortran 90 and making use of the module STUFF, which gives meaning to the kind dbl. There is a number of options in the subroutine. In using the subroutine to generate Tables A.3–A.5, the recommendations of Whiting and Carr [571] were followed, setting both parameters α and β to zero, and not including the boundary points indexed zero and $N+1$. See the book by Villadsen and Michelsen for the details.

C.3 Example Programs

Program COTT_EX

This program simulates the Cottrell experiment, as discussed in Chap. 5. The output, upon running this for NT=100 and lambda=0.45, is

```
iT     T       G   G(analyt)
 1   0.010   4.427   5.642
 2   0.020   3.921   3.989
 4   0.040   2.808   2.821
 8   0.080   1.996   1.995
16   0.160   1.414   1.410
```

```
 32   0.320   0.999   0.997
 64   0.640   0.706   0.705
100   1.000   0.565   0.564
```

Ordinarily, there would be some header information, echoing the input data and data derived from it. Also, the prompts produced by the program have been omitted.

Program `CHRONO_EX`

This program is much the same as `COTT_EX`, but with the derivative boundary condition; see Chap. 5 for background information. Note that C_0 is calculated once at the start, so that it conforms to the rest of the initial concentration profile, so as to satisfy the boundary condition. This calculation is repeated after every run of the innermost loop in which all concentrations $C_1, \ldots, C_N$ are recalculated. In this way, the old concentrations always include the proper C_0 value. This must always be the case, no matter what the boundary conditions are.

This produces, again for `NT`=100 and `lambda`=0.45, the (trimmed) output

```
iT      T      C(0)  C(0)(analyt)
 1   0.010   0.887   0.900
 2   0.020   0.849   0.859
 4   0.040   0.792   0.800
 8   0.080   0.711   0.717
16   0.160   0.596   0.600
32   0.320   0.431   0.434
64   0.640   0.198   0.200
100  1.000  -0.002   0.000
```

Program `CV_EX`

This is a simple simulation of a CV experiment, using the explicit method EX, and assuming a quasireversible reaction,

$$A + e^- \leftrightarrow B \tag{C.5}$$

with dimensionless heterogeneous rate constant K_0, as defined in (2.28). If this constant is input as $K0 > 1000$, the system is assumed reversible and the Nernst boundary condition is applied. See Sect. 5.5 for the details of how to apply the boundary conditions. Here, an important point needs to be made. When carrying out a step forward in time using method EX, the old C-arrays are used to explicitly generate the new arrays at the new time. Again, the old C-arrays must include the boundary values corresponding to the arrays, so that the boundary conditions hold.

The program draws on several subroutines and functions, already described, such as GOFUNC, GOBETA and MINMAX to calculate peak and trough currents and at what potentials they occur.

This produces the following example output in a particular run:

```
CV_EX
Lambda   =     0.450
H        =     0.149
nT per p =   100
N        =   278
K0       =  1001.000
 1190 points written into plot file.
Top current and -p =  0.4463 -1.1074
Bot current and -p = -0.3335  1.1318
```

Being an explicit method, the program uses 278 points in space, even though H is rather large at 0.149. The peak current is surprisingly accurate (to all 4 decimals) but the peak potential is not (it should be -1.1090 for the K_0 value set here at 1001 to force the Nernst boundary condition). Generally, one finds that the peak potential is a more sensitive indicator of how well a given simulation method works.

Program COTT_CN

This program does the same work as the earlier one, COTT_EX, but uses Crank-Nicolson (with equal intervals). It also includes the choice of M Pearson substeps within the first step, to damp the oscillations, as discussed in Chap. 8, Sect. 8.5.1.

Using similar parameters as for the earlier program, but making use of CN's stability with respect to λ, that parameter is set to 3, giving more points in the X-range. Here is a sample output from a run (again omitting the dialog part):

```
CN Cottrell simulation.
NT     =   100
Lambda =     3.00
N      =   104 pts along X.
10 Pearson substeps within first step.

   iT       T       Gsim     log(err)
    2     0.020     3.878    -1.556
    4     0.040     2.802    -2.164
    8     0.080     1.992    -2.868
   16     0.160     1.410    -3.845
   32     0.320     0.997    -4.163
   64     0.640     0.705    -4.149
  100     1.000     0.564    -4.268
```

Note the improved accuracy, using 10 Pearson steps, despite using only 100 steps in time. The explicit program gave rise to errors in the third decimal at the end of the simulation, but here they lie at around the fourth or better.

Program CHRONO_CN

A chronopotentiometry program, using CN, is shown here, again, as with the above COTT_CN, with equal intervals. The two programs are in fact very similar, differing only in the boundary conditions in the CN routine, and the initialisation. As before, old known concentrations always include a conforming C_0.

Note also that although the Pearson option has been included, it is not really needed here. CN does not oscillate with the constant current start. A sample output follows, again using $\lambda = 3$ as for Cottrell above.

```
CN chronopot. simulation.
NT     =   100
Lambda =     3.00
N      =   104 pts along X.
 1 Pearson substeps within first step.

   iT       T       C(0)     log(err)
    2     0.020     0.859    -8.414
    4     0.040     0.800    -7.937
    8     0.080     0.717    -7.972
   16     0.160     0.600    -8.417
   32     0.320     0.434    -8.874
   64     0.640     0.200    -9.284
  100     1.000     0.000    -9.531
```

As with Cottrell, note the accuracy compared with CHRONO_EX. Note also that a single "Pearson" step has been used, meaning no subdivision of the first step.

Program LSV_CN

The following shows a CN program with unequal intervals, simulating a single LSV sweep for a reversible system. Apart from writing out the (G, T) results for plotting, the program also detects the peak value and the potential at which it peaks. The boundary condition part is described in Chap. 6, page 93.

An example output, apart from the data file for plotting, was

```
LSV_CN_UN:
nT     =   100 per p-unit
pstart =    12.000
pstop  =   -12.000
```

```
X(1)   =     0.00100
N      =    50
Xlim   =    29.394
G, using     3 points.
gamma  =     1.18811 (found by iteration)

G-peak of    0.4472 found at p = -1.1091
```

This program uses exponentially expanding intervals in X, and we started here with a first interval of 0.001 and demanded 50 points across the diffusion space. This set the γ value as seen above. The peak current is a little off (it should be 0.4463) but the peak potential is quite good (the exact value is -1.1090).

Program COTT_EXTRAP

This is an example of a Cottrell simulation using second-order extrapolation based on the BI (Laasonen) method and unequal intervals. Three-point spatial discretisation is used here.

Program COTT_EXTRAP4

This program is again a Cottrell simulation using second-order extrapolation based on the BI (Laasonen) method and unequal intervals, but in contrast with the above program COTT_EXTRAP, this one makes use of the four-point spatial derivative approximation, and the GU-function. It performs a little better than the above program, at little extra programming effort.

A Nonlinear System Linearised: Program BP_LIN

One of the problems mentioned in Chap. 8 is that of second-order homogeneous chemical reactions, which give rise to nonlinear terms in the transport equations. One such system is the Birk-Perone reaction [121,146], in which a light flash produces an electroactive substance in solution, which decays with a second-order reaction while it is electrolysed. If CN is used to simulate this, the term in C_i^2 can be linearised to a good, second-order approximation. When one does not choose or is prevented from linearisation, a Newton approach, as described in that chapter, must be used. The two programs are examples of both approaches. The same system is simulated; a Cottrell experiment (potential jump) on a decaying substance, both using CN. The first program uses linearisation, the second the Newton method. The two programs produce almost, but not quite the same, results, the Newton version being slightly more accurate. Both programs make automatic use of the Pearson start, by subdividing the first step in that number of substeps that gives a unity value to $\delta T/H_1^2$. In this way, negative concentrations are avoided.

A Nonlinear System Using Newton: Program BP_NONLIN

This is the nonlinear version using Newton iterations.

A Nonlinear System Using Rosenbrock: Program BP_ROS

Rosenbrock methods, as discussed in Chaps. 4 and 9, have in a sense an inbuilt Newton iteration, and thus suggest themselves for the solution of nonlinear systems. The program BP_ROS, like the two previous ones, simulates the Birk and Perone system, using one of two Rosenbrock methods. The program illustrates not only the nonlinear handling, but also the handling of time-dependent systems, as this one is, due to the decaying outer boundary, C_{N+1}. There is an analytical solution for this value and its derivative, and both are made use of in the program. It works about as well as the above two and illustrates the Rosenbrock approach. The program, besides the usual STUFF, also makes use of the special Rosenbrock module ROSTUFF, see above. The module contains the subroutine ROCOEFFS, which sets the Rosenbrock constants. Only the Rosenbrock variants ROS2 and ROWDA3 are allowed for in the program. The inconsistency at $T = 0$, mentioned in Sect. 9.4, is overcome by the simple trick of setting C_0 to zero initially.

EC Reaction, Cyclic Voltammetry: CV_EC

The program CV_EC uses CN to simulate the rather simple EC reaction as described in Chap. 8. There are no complications here. The output is in the form of a data file for plotting. It makes use of the simple subroutine MAT_INV_22 to solve the small 2×2 system of equations for the boundary values. This could have been done directly but when we already have this routine, why not use it? Note that the a' coefficients are different for the two species but are constant throughout the simulation and are therefore precalculated. This system is not coupled, so that the (scalar) Thomas algorithm can be used. Figure C.1 shows the result of some runs of this program. The fat curve is for $K = 0$, that is, plain reversible CV without a chemical reaction, and the numbers marked on the curves show the K values input to the program. As expected, as K increases, the negative-going peak (at the top) shifts in the positive direction and the trough (on the reverse sweep) becomes smaller, to disappear entirely for large K.

CV of the EC′ Reaction: Program CV_RUCAT

The coupled system arising from the EC′ or catalytic system, described in Chap. 8, was programmed using CN and the Rudolph method. That is, the two concentration vectors were gathered into a vector of two-element vectors, and the usual coefficients in the discrete system of equations become a number

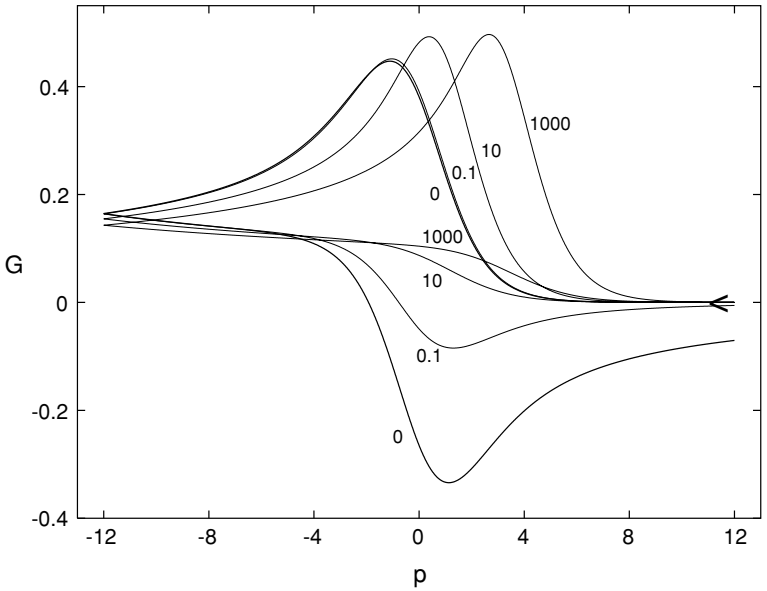


Fig. C.1. CV of the EC reaction, K values as marked

of coefficient matrices. For this system, the $\boldsymbol{A}$ matrices are constant over the whole simulation and can be precomputed. In fact, one finds that one needs not these but the inverse of the $\boldsymbol{A}'$ matrices that are the result of reducing the system to two variable (vectors) for each row; and it is these that are precomputed and stored. The program outputs a data file for plotting. As with the program `CVEC`, the routine `MAT_INV_22` is used, more extensively here. Figure C.2 shows a family of CV curves, the fat curve again being that for the plain reversible case, and the others with rate constants K as marked. As K increases, the curve becomes increasingly S-shaped, with a plateau of height equal to $\sqrt{K}$.

LSV Simulation with iR Drop and Capacitance: Program `LSV4IRC`

The program `LSV4IRC` is a simulation of a reversible reaction with input values of ρ (dimensionless uncompensated resistance and γ_c (dimensionless double layer capacity). Unequal intervals are used, with asymmetric 4-point second spatial derivatives, and second order extrapolation in the time direction. The nonlinear set of 6 equations for the boundary values is solved by Newton-Raphson iteration. Some results are seen in Chap. 11.

Program `UMDE_DIRECT`

This program simulates a Cottrellian potential jump at a UMDE, using three-point BDF for the time integration (starting it with a single BI step), and

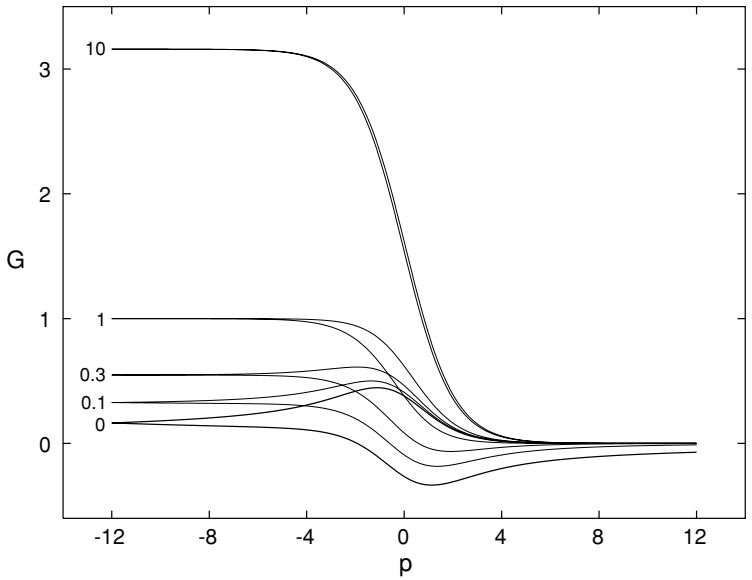


Fig. C.2. CV of the catalytic (EC′) reaction, K values as marked

four-point asymmetric spatial discretisation, of the form $u_2''(4)$, on an (R, Z) grid with exponentially expanding intervals expanding upwards in Z, and away from the electrode edge at $R = 1$ in both directions. This is discussed in Chap. 12, Sect. 12.3.1. The sparse solver `MA28` is used.

Program `UMDE_VB`

This is a version of the disk simulator but using a grid in the conformal space given by the Verbrugge/Baker transformation, as discussed in Chap. 12, page 224.

Program `CHANNEL_BAND`

This program does a steady state simulation of the current at a narrow band electrode at the bottom of a channel with laminar flow of electrolyte through it, as discussed in Chap. 13, page 241. It is done as a march along X, the direction of flow.

References

1. http://hsl.rl.ac/archive/hslarchive.html
2. http://www.netlib.org
3. http://www.netlib.org/ode/vode.f
4. http://www.cs.berkeley.edu/~kdatta/lsode/lsode.html
5. http://db.cwi.nl/rapporten/abstract.php?abstractnr=892
6. http://www.engineering.ucsb.edu/~cse/software.html
7. http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/phreeqc/
8. http://www.mgnet.org/mgnet-codes-pltmg.html
9. http://www.comsol.com
10. http://www.simulog.fr/eis/2flux1.htm
11. http://www.orcad.com
12. http://www.cyf-kr.edu.pl/~nbbienia/elsim3ad.html
13. http://www.bioanalytical.com
14. http://www.digielch.de
15. http://deepspace9.sci.ccny.cuny.edu/CVSIM.HTML
16. http://www.DrHuang.com
17. http://dbweb.liv.ac.uk/ltsnpsc/swrevs/5polar.htm
18. http://sourceforge.net/projects/echempp
19. http://physchem.ox.ac.uk/~rgc/F11/index.html
20. http://www.kanazawa-bidai.ac.jp/~momo/qrcv/QRCV.html
21. http://lem.ch.unito.it/chemistry/esp_manual.html
22. http://www.finishing.com/Software/elsyca.html
23. http://colossus.chem.umass.edu/bvining/free.htm
24. http://www.simtel.net/product.php?url_fb_product_page=12300
25. http://www.condecon.com/condesim.htm
26. http://transient.mkolar.org/
27. Ablow C.M., Schechter S., J. Comp. Phys. **27**, 351–362 (1978)
28. Abramowitz M., Stegun I.A. (Eds.), *Handbook of Mathematical Functions*, Dover Publications, New York (1969)
29. Albery W.J., Trans. Faraday Soc. **62**, 1915–1919 (1966)
30. Albery W.J., Brett C.M.A., J. Electroanal. Chem. **148**, 201–210 (1983)
31. Albery W.J., Bruckenstein S., Trans. Faraday Soc. **62**, 1920–1931 (1966)
32. Albery W.J., Bruckenstein S., Trans. Faraday Soc. **62**, 1946–1954 (1966)
33. Albery W.J., Bruckenstein S., Johnson D.C., Trans. Faraday Soc. **62**, 1938–1945 (1966)
34. Albery W.J., Bruckenstein S., Napp D.T., Trans. Faraday Soc. **62**, 1932–1937 (1966)
35. Albery W.J., Chadwick A.T., Coles B.A., Hampson N.A., J. Electroanal. Chem. **75**, 229–239 (1977)

36. Albery W.J., Compton R.G., Chadwick A.T., Coles B.A., Lenkaits J.A., J. Chem. Soc. Faraday Trans. I **76**, 1391–1401 (1980)
37. Albery W.J., Jones C.C., Mount A.R., in: *Comprehensive Chemical Kinetics* (Edited by R.G. Compton, G. Hancock), Elsevier, Amsterdam, vol. 29, 129–148 (1989)
38. Alden J.A., Booth J., Compton R.G., Dryfe R.A.W., Sanders G.H.W., J. Electroanal. Chem. **389**, 45–54 (1995)
39. Alden J.A., Compton R.G., J. Electroanal. Chem. **404**, 27–35 (1996)
40. Alden J.A., Compton R.G., J. Electroanal. Chem. **415**, 1–12 (1996)
41. Alden J.A., Compton R.G., J. Phys. Chem. B **101**, 9606–9616 (1997)
42. Alden J.A., Feldman M.A., Hill E., Prieto F., Oyama M., Coles B.A., Compton R.G., Anal. Chem. **70**, 1707–1720 (1998)
43. Alden J.A., Hutchinson F., Compton R.G., J. Phys. Chem. B **101**, 949–958 (1997)
44. Alhumaizi K., Comp. Chem. Eng. **28**, 1759–1769 (2004)
45. Altas I., Dym J., Gupta M.M., Manohar R.P., SIAM Journal on Scientific Computing (1996 preprint), http://na.cs.yale.edu/mgnet/www/mgnet/papers/Altas-Dym-Gupta-Manohar/high-order.ps.gz
46. Amatore C., in: *Physical Electrochemistry* (Edited by I. Rubinstein), Marcel Dekker, New York, 131–208 (1995)
47. Amatore C., Oleinick A., Svir I., Electrochem. Commun. **5**, 989–994 (2003)
48. Amatore C., Oleinick A., Svir I., J. Electroanal. Chem. **564**, 245–260 (2004)
49. Amatore C., Oleinick A.I., Svir I.B., J. Electroanal. Chem. **553**, 49–61 (2003)
50. Amatore C., Savéant R.G., J. Electroanal. Chem. **102**, 21–40 (1979)
51. Amatore C.A., Deakin M.R., Wightman R.M., J. Electroanal. Chem. **206**, 23–36 (1986)
52. Amatore C.A., Fosset B., J. Electroanal. Chem. **328**, 21–32 (1992)
53. Amphlett J.L., Denuault G., J. Phys. Chem. B **102**, 9946–9951 (1998)
54. Andersen J.L., Moldoveanu S., J. Electroanal. Chem. **179**, 107–117 (1984)
55. Ang K.P., Gunasingham H., Tay B.T., J. Singapore Nat. Acad. Sci. **16**, 80–86 (1987)
56. Aoki K., J. Electroanal. Chem. **284**, 35–42 (1990)
57. Aoki K., Electroanalysis **5**, 627–639 (1993)
58. Aoki K., Akimoto K., Tokuda K., Matsuda H., Osteryoung J., J. Electroanal. Chem. **171**, 219–230 (1984)
59. Aoki K., Morita M., Niwa O., Tabei H., J. Electroanal. Chem. **256**, 269–282 (1988)
60. Aoki K., Nishiki Y., J. Appl. Electrochem. **19**, 183–187 (1989)
61. Aoki K., Osteryoung J., J. Electroanal. Chem. **122**, 19–35 (1981)
62. Aoki K., Osteryoung J., J. Electroanal. Chem. **160**, 335–339 (1984)
63. Aoki K., Tanaka M., J. Electroanal. Chem. **266**, 11–20 (1989)
64. Aoki S., Kishimoto K., Miyasaka M., Corrosion **44**, 926–932 (1988)
65. Arkoub I.A., Amatore C., Sella C., Thouin L., Warkocz J.S., J. Phys. Chem. B **105**, 8694–8703 (2001)
66. Atkins P.W., *Physical Chemistry*, Oxford University Press, Oxford, UK, sixth ed. (1998)
67. Bacha S., Bergel A., Comtat M., J. Electroanal. Chem. **359**, 21–38 (1993)
68. Balslev H., Britz D., Acta Chem. Scand. **46**, 949–955 (1992)
69. Banks C.E., Compton R.G., Fisher A.C., Henley I.E., Phys. Chem. Chem. Phys. **6**, 3147–3152 (2004)

70. Barakat H.Z., Clark J.A., Trans. ASME J. Heat Transfer 421–427 (1966)
71. Bard A.J., Crayston J.A., Kittlesen G.P., Shea T.V., Wrighton M.S., Anal. Chem. **58**, 2321–2331 (1986)
72. Bard A.J., Denuault G., Friesner R.A., Dornblaser B.C., Tuckerman L.S., Anal. Chem. **63**, 1282–1288 (1991)
73. Bard A.J., Faulkner L.R., *Electrochemical Methods*, John Wiley, New York (1980)
74. Bard A.J., Faulkner L.R., *Electrochemical Methods*, John Wiley, New York (2001)
75. Barker A.L., Macpherson J.V., Slevin C.J., Unwin P.R., J. Phys. Chem. B **102**, 1586–1598 (1998)
76. Barker A.L., Unwin P.R., J. Phys. Chem. B **105**, 12019–12031 (2001)
77. Barker A.L., Unwin P.R., Amemiya S., Zhou J., Bard A.J., J. Phys. Chem. B **103**, 7260–7269 (1999)
78. Barker A.L., Unwin P.R., Zhang J., Electrochem. Commun. **3**, 372–378 (2001)
79. Barker P.D., Hill H.A.O., Walton N.J., J. Electroanal. Chem. **260**, 303–326 (1989)
80. Bartolini R., Fantini L., Gallone P., Ann. Chim. (Rome) **66**, 7–18 (1976)
81. Basak J., Penar J., Sykut K., Ann. Univ. Mariae Curie - Skłodowska, Sectio AA **XLII/XLIII**, 43–49 (1987/1988)
82. Basha C.A., Sangaranarayanan M.V., J. Electroanal. Chem. **261**, 431–436 (1989)
83. Bauer H.H., *Electrodics*, Thieme, Stuttgart (1972)
84. Baur J.E., Motsegood P.N., J. Electroanal. Chem. **572**, 29–40 (2004)
85. Bellamy A.J., Howat G., MacKirdy I., J. Chem. Soc., Perkin **11**, 786–793 (1978)
86. Bernstein C., Heindrichs A., Vielstich W., J. Electroanal. Chem. **87**, 81–90 (1978)
87. Bianchi F., Ferrigno R., Girault H.H., Anal. Chem. **72**, 1987–1993 (2000)
88. Bickley W.G., Math. Gaz. **25**, 19–27 (1941)
89. Bieniasz L.K., Comput. Chem. **16**, 311–317 (1992)
90. Bieniasz L.K., Comput. Chem. **16**, 11–14 (1992)
91. Bieniasz L.K., J. Electroanal. Chem. **347**, 15–30 (1993)
92. Bieniasz L.K., Comput. Chem. **17**, 355–368 (1993)
93. Bieniasz L.K., J. Electroanal. Chem. **360**, 119–138 (1993)
94. Bieniasz L.K., J. Electroanal. Chem. **345**, 13–25 (1993)
95. Bieniasz L.K., J. Electroanal. Chem. **374**, 1–22 (1994)
96. Bieniasz L.K., J. Electroanal. Chem. **374**, 23–35 (1994)
97. Bieniasz L.K., J. Electroanal. Chem. **379**, 71–87 (1994)
98. Bieniasz L.K., J. Electroanal. Chem. **404**, 195–208 (1996)
99. Bieniasz L.K., Comput. Chem. **21**, 1–12 (1997)
100. Bieniasz L.K., J. Electroanal. Chem. **469**, 97–115 (1999)
101. Bieniasz L.K., J. Electroanal. Chem. **481**, 115–133 (2000), corrigendum: *ibid.* **565** (2004) 131
102. Bieniasz L.K., J. Electroanal. Chem. **481**, 134–151 (2000), corrigendum: *ibid.* **565** (2004) 133
103. Bieniasz L.K., Electrochem. Commun. **3**, 149–153 (2001)
104. Bieniasz L.K., in: *Modern Aspects of Electrochemistry* (Edited by B.E. Conway, R.E. White), Kluwer/Plenum, New York, vol. 35, 135–195 (2002)

105. Bieniasz L.K., J. Electroanal. Chem. **527**, 21–32 (2002), corrigendum: *ibid.* **565** (2004) 141
106. Bieniasz L.K., Comput. Chem. **26**, 633–644 (2002)
107. Bieniasz L.K., J. Electroanal. Chem. **558**, 167–170 (2003)
108. Bieniasz L.K., Comp. Biol. Chem. **27**, 315–325 (2003)
109. Bieniasz L.K., J. Comput. Chem. **25**, 1515–1521 (2004)
110. Bieniasz L.K., J. Comp. Chem. **25**, 1075–1083 (2004)
111. Bieniasz L.K., Britz D., Acta. Chem. Scand. **47**, 757–767 (1993)
112. Bieniasz L.K., Britz D., Anal. Chim. Acta **278**, 59–70 (1993)
113. Bieniasz L.K., Britz D., J. Electroanal. Chem. **503**, 141–152 (2001)
114. Bieniasz L.K., Britz D., Pol. J. Chem. **78**, 1195–1219 (2004)
115. Bieniasz L.K., Bureau C., J. Electroanal. Chem. **481**, 152–167 (2000), corrigendum: *ibid.* **565** (2004) 135
116. Bieniasz L.K., Østerby O., Britz D., Comput. Chem. **19**, 121–136 (1995)
117. Bieniasz L.K., Østerby O., Britz D., Comput. Chem. **19**, 351–355 (1995)
118. Bieniasz L.K., Østerby O., Britz D., Comput. Chem. **19**, 357–370 (1995)
119. Bieniasz L.K., Østerby O., Britz D., Comput. Chem. **21**, 391–401 (1997)
120. Bird R.B., Steward W.E., Lightfoot E.N., *Transport Phenomena*, John Wiley, NY (1960)
121. Birk J.R., Perone S.P., Anal. Chem. **40**, 496–500 (1968)
122. Blom J.G., Sanz-Serna J.M., Verwer J.G., J. Comp. Phys. **74**, 191–213 (1988)
123. Bond A.M., Feldberg S.W., J. Phys. Chem. B **102**, 9966–9974 (1998)
124. Bond A.M., Mahon P.J., J. Electroanal. Chem. **439**, 37–53 (1997)
125. Bond A.M., Oldham K.B., Zoski C.G., J. Electroanal. Chem. **245**, 71–104 (1988)
126. Borkowski M., Stojek Z., Electroanalysis **4**, 615–621 (1992)
127. Bortels L., Deconinck J., Bossche B.V.D., J. Electroanal. Chem. **404**, 15–26 (1996)
128. Bowyer W.J., Engelman E.E., Evans D.H., J. Electroanal. Chem. **262**, 67–82 (1989)
129. Brenan K., IEEE Trans. Aut. Control **AC-31**, 266–269 (1986)
130. Brenan K.E., Campbell S.L., Petzold L.R., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia (1996)
131. Britz D., J. Electroanal. Chem. **88**, 309–352 (1978)
132. Britz D., Electrochim. Acta **25**, 1449–31452 (1980)
133. Britz D., Anal. Chim. Acta **193**, 277–285 (1987)
134. Britz D., *Digital Simulation in Electrochemistry*, Springer, Berlin (1988)
135. Britz D., J. Electroanal. Chem. **240**, 17–26 (1988)
136. Britz D., Anal. Chem. **66**, 792A–793A (1994), review of David K. Gosser, "Cyclic Voltammetry: Simulation and Analysis of Reaction Mechanisms".
137. Britz D., Anal. Chem. **67**, 600A–601A (1995), review of DigiSim.
138. Britz D., J. Electroanal. Chem. **406**, 15–21 (1996)
139. Britz D., BIT **38**, 217–218 (1998)
140. Britz D., Comput. Chem. **22**, 237–243 (1998)
141. Britz D., Comput. Chem. Eng. **23**, 297–300 (1999)
142. Britz D., J. Electroanal. Chem. **515**, 1–7 (2001)
143. Britz D., Electrochem. Commun. **5**, 195–198 (2003)
144. Britz D., da Silva B.M., Avaca L.A., Gonzales E.R., Anal. Chim. Acta **239**, 87–93 (1990)

145. Britz D., Heinze J., Mortensen J., Störzbach M., J. Electroanal. Chem. **240**, 27–43 (1988)
146. Britz D., Kastening B., J. Electroanal. Chem. **56**, 73–90 (1974)
147. Britz D., Nielsen M.F., Coll. Czech. Chem. Commun. **56**, 20–41 (1991)
148. Britz D., Østerby O., J. Electroanal. Chem. **368**, 143–147 (1994)
149. Britz D., Østerby O., Strutwolf J., Comput. Biol. Chem. **27**, 253–263 (2003)
150. Britz D., Østerby O., Strutwolf J., Svennesen T.K., Comput. Chem. **26**, 97–103 (2002)
151. Britz D., Poulsen K., Strutwolf J., Electrochim. Acta **50**, 107–113 (2004)
152. Britz D., Strutwolf J., Comput. Chem. **24**, 673–684 (2000)
153. Britz D., Strutwolf J., Comp. Biol. Chem. **27**, 327—337 (2003)
154. Britz D., Strutwolf J., Thøgersen L., J. Electroanal. Chem. **512**, 119–123 (2001)
155. Britz T.J., Britz D., J. Electroanal. Chem. **546**, 123–125 (2003)
156. Brookes B.A., Davies T.J., Fisher A.C., Evans R.G., Wilkins S.J., Yunus K., Wadhawan J.D., Compton R.G., J. Phys. Chem. B **107**, 1616–1627 (2003)
157. Bruckenstein S., Johnson D.C., Anal. Chem. **36**, 2186–2187 (1964)
158. Brunner E., Z. phys. Chem. **47**, 56–102 (1904)
159. Cabán R., Chapman T.W., J. Electrochem. Soc. **123**, 1036–1041 (1976)
160. Caillaud J.B., Padmanabhan L., Chem. Eng. J. **2**, 227–232 (1971)
161. Carnahan B., Luther H.A., Wilkes J.O., *Applied Numerical Methods*, John Wiley, NY (1969)
162. Chen T., Dong S., Xie Y., J. Electroanal. Chem. **379**, 239–245 (1994)
163. Cheney W., Kincaid D., *Numerical Mathematics and Computing*, Brooks/Cole, Belmont, California (1985)
164. Clarenbach S., Grabner E.W., Ber. Bunsenges. phys. Chem. **80**, 115–121 (1976)
165. Clarenbach S., Grabner E.W., Brauer E., Ber. Bunsenges. phys. Chem. **77**, 908–913 (1973)
166. Cochran W.G., Proc. Cambr. Phil. Soc. **30**, 365–375 (1934)
167. Coen S., Cope D.K., Tallman D.E., J. Electroanal. Chem. **215**, 29–48 (1986)
168. Coles B.A., Compton R.G., Brett C.M.A., Brett A.M.C.F.O., J. Electroanal. Chem. **381**, 99–104 (1995)
169. Collatz L., Schriften Math. Sem. Inst. ang. Math. Univ. Berlin **3**, 1–35 (1935)
170. Collatz L., *Numerische Behandlung von Differentialgleichungen*, Springer Verlag, Heidelberg (1960)
171. Compton R.G., Coles B.A., Fisher A.C., J. Phys. Chem. **98**, 2441–2445 (1994)
172. Compton R.G., Coles B.A., Gooding J.J., Fisher A.C., J. Phys. Chem. **98**, 2446–2451 (1994)
173. Compton R.G., Dryfe R.A.W., Wellington R.G., Hirst J., J. Electroanal. Chem. **383**, 13–19 (1995)
174. Compton R.G., Pilkington M.B.G., Stearn G.M., J. Chem. Soc., Faraday Trans. I **84**, 2155–2171 (1988)
175. Cooper J.A., Compton R.G., Electroanalysis **10**, 141–155 (1998)
176. Cope D.K., Scott C.H., Kalapathy U., Tallman D.E., J. Electroanal. Chem. **280**, 27–35 (1990)
177. Cope D.K., Scott C.H., Tallman D.E., J. Electroanal. Chem. **285**, 49–69 (1990)
178. Cope D.K., Tallman D.E., J. Electroanal. Chem. **285**, 85–92 (1990)
179. Cope D.K., Tallman D.E., J. Electroanal. Chem. **285**, 79–84 (1990)

180. Cope D.K., Tallman D.E., Electrochem. Soc. Proc. **99-5**, 82–89 (1999)
181. Cottrell F.G., Z. phys. Chem. **42**, 385–431 (1903)
182. Courant R., Friedrichs K., Lewy H., Math. Ann. **100**, 32–74 (1928)
183. Crank J., *The Mathematics of Diffusion*, Clarendon Press, Oxford, 2 ed. (1975)
184. Crank J., Furzeland R.M., J. Inst. Maths. Applics **20**, 355–370 (1977)
185. Crank J., Nicolson P., Proc. Cambridge Phil. Soc. **43**, 50–67 (1947), reprinted in Adv. Comp. Math. 6 (1996) 207–226, with some slight changes to the list of references
186. Crowder H.J., Dalton C., J. Comp. Phys. **7**, 32–45 (1971)
187. Cryer C.W., BIT **12**, 17–25 (1972)
188. Curtiss C.F., Hirschfelder J.O., Proc. Nat. Acad. Sci. US **38**, 235–243 (1952)
189. Dahlquist G.G., BIT **3**, 27–43 (1963)
190. Dahlquist G.G., Proc. Symp. Appl. Math. **15**, 147–158 (1963)
191. Damaskin B.B., Petrii O.A., Batrakov V.V., *Adsorption organischer Verbindungen an Elektroden*, Akademie-Verlag, Berlin (1975)
192. Dan C., Van den Bossche B., Bortels L., Nelissen G., Deconinck J., J. Electroanal. Chem. **505**, 12–23 (2001)
193. Davies T.J., Brookes B.A., Compton R.G., J. Electroanal. Chem. **566**, 193–216 (2004)
194. Davies T.J., Brookes B.A., Fisher A.C., Yunus K., Wilkins S.J., Greene P.R., Wadhawan J.D., Compton R.G., J. Phys. Chem. B **107**, 6431–6449 (2003)
195. de Boor C., in: *Conference on the Numerical Solution of Differential Equations, Dundee, Scotland 1973.* (Edited by G.A. Watson), Springer, Berlin (1974), 12–20
196. de Swart J.J.B., Blom J.G., *Experiences with sparse matrix solvers in parallel ODE software*, Tech. Rep. Rept. NM-R9520 1995, Centrum voor Wiskunde en Informatica (CWI), Amsterdam (1995), accessible at http://db.cwi.nl/rapporten/abstract.php?abstractnr=892
197. Deakin M.R., Wightman R.M., Amatore C.A., J. Electroanal. Chem. **215**, 49–61 (1986)
198. Deconinck J., Magetto G., Vereecken J., J. Electrochem. Soc. **132**, 2960–2965 (1985)
199. Delahay P., J. Am. Chem. Soc. **75**, 1190–1196 (1953)
200. Delahay P., *Double Layer and Electrode Kinetics*, Interscience, New York (1966)
201. Delahay P., Mohilner D.M., J. Am. Chem. Soc. **84**, 4247–4252 (1962)
202. Delahay P., Stiehl G.L., J. Am. Chem. Soc. **74**, 3500–3505 (1952)
203. Delahay P., Trachtenberg I., J. Am. Chem. Soc. **79**, 2355–2362 (1957)
204. Demaille C., Unwin P.R., Bard A.J., J. Phys. Chem. **100**, 14137–14143 (1996)
205. Deng Z., Lin X., Chin. J. Anal. Chem. **27**, 1376–1380 (1999), [In Chinese, Engl. abstract]
206. Deng Z.X., Lin X.Q., J. Electroanal. Chem. **464**, 215–221 (1999)
207. Deng Z.X., Lin X.Q., Tong Z.H., Chin. J. Chem. **20**, 252–262 (2002)
208. Deng Z.X., Lin X.Q., Tong Z.H., Chin. J. Chem. **21**, 1137–1145 (2003)
209. Deng Z.X., Tong Z.H., Lin X.Q., J. Electroanal. Chem. **568**, 235–245 (2004)
210. Deng Z.X., X.-Q-Lin, Tong Z.H., Chin. J. Chem. **22**, 719–726 (2004)
211. Diehl H., Biggs D.L., Talanta **30**, 894–898 (1983)
212. Dillard J.W., Turner J.A., Osteryoung R.A., Anal. Chem. **49**, 1246–1250 (1977)

213. Dorfi E.A., Drury L.O., J. Comp. Phys. **69**, 175–195 (1987)
214. Douglas Jr J., Gallie Jr T.M., Proc. Am. Math. Soc. **6**, 787–793 (1955)
215. Duff I.S., Reid J.K., ACM Trans. Math. Soft. **5**, 18–35 (1979)
216. DuFort E.C., Frankel S.P., Math. Tables Aids Comput. **7**, 135–152 (1953)
217. Eddowes M.J., J. Electroanal. Chem. **159**, 1–22 (1983)
218. Emmons H.W., Quart. Appl. Math. **2**, 173–195 (1944)
219. Engblom S.O., Cope D.K., Tallman D.E., J. Electroanal. Chem. **406**, 23–31 (1996)
220. Engeln-Müllges G., Uhlig F., *Numerical Algorithms with Fortran*, Springer-Verlag, Berlin Heidelberg (1996)
221. Engstrom R.C., Pharr C.M., Koppang M.D., J. Electroanal. Chem. **221**, 251–255 (1997)
222. Engstrom R.C., Weber M., Wunder D.J., Burgess R., Winquist S., Anal. Chem. **58**, 844–848 (1986)
223. Evans D.J., Abdullah A.R.B., Int. J. Comp. Math. **14**, 73–105 (1983)
224. Evans N.T.S., Gourlay A.R., J. Inst. Maths. Appl. **19**, 239–251 (1977)
225. Eyres N.R., Hartree D.R., Ingham J., Jackson R., Sarjant R.J., Wagstaff J.B., Phil. Trans. Roy. Soc. Lond. A **240**, 1–57 (1946)
226. Fanelli N., Záliš S., Pospíšil C., J. Electroanal. Chem. **262**, 35–44 (1989)
227. Fanelli N., Záliš S., Pospíšil C., J. Electroanal. Chem. **288**, 263–269 (1990)
228. Fang H., Chen H.Y., Chin. J. Chem. **15**, 250–259 (1997)
229. Feldberg S.W., in: *Electroanal. Chem.* (Edited by A.J. Bard), Marcel Dekker, New York, vol. 3, 199–296 (1969)
230. Feldberg S.W., J. Electroanal. Chem. **109**, 69–82 (1980)
231. Feldberg S.W., J. Electroanal. Chem. **127**, 1–10 (1981)
232. Feldberg S.W., J. Electroanal. Chem. **222**, 101–106 (1987)
233. Feldberg S.W., J. Electroanal. Chem. **290**, 49–65 (1990)
234. Feldberg S.W., Auerbach C., Anal. Chem. **36**, 505–509 (1964)
235. Feldberg S.W., Bowers M.L., Anson F.C., J. Electroanal. Chem. **215**, 11–28 (1986)
236. Feldberg S.W., Goldstein C.I., J. Electroanal. Chem. **397**, 1–10 (1995)
237. Fernández J.L., Bard A.J., Anal. Chem. **76**, 2281–2289 (2004)
238. Ferrigno R., Brevet P.F., Girault H.H., J. Electroanal. Chem. **430**, 235–242 (1997)
239. Ferrigno R., Brevet P.F., Girault H.H., Electrochim. Acta. **42**, 1895–1903 (1997)
240. Ferrigno R., Girault H.H., J. Electroanal. Chem. **492**, 1–6 (2000)
241. Ferrigno R., Josserand J., Brevet P.F., Girault H.H., Electrochim. Acta **44**, 587–595 (1998)
242. Fick A., Pogg. Ann. **94**, 59–86 (1855)
243. Finkley H.O., in: *Electroanal. Chem.* (Edited by A.J. Bard, I. Rubinstein), Marcel Dekker, New York, vol. 19, 109–335 (1996)
244. Fisher A.C., Compton R.G., J. Phys. Chem. **95**, 7538–7542 (1991)
245. Fisher A.C., Compton R.G., Electroanal. **4**, 311–315 (1992)
246. Flanagan J.B., Marcoux L., J. Phys. Chem. **77**, 1051–1055 (1973)
247. Flanagan J.B., Marcoux L., J. Phys. Chem. **78**, 718–723 (1974)
248. Flanagan J.B., Takahashi K., Anson F.C., J. Electroanal. Chem. **81**, 261–273 (1977)
249. Fleischmann M., Pons S., Rolison D.R., Schmidt P.P., *Ultramicroelectrodes*, Datech Systems Inc., Morganton, NC (1987)

250. Fletcher C.A.J., *Computational Techniques for Fluid Dynamics, Volume I*, Springer, Berlin, 2 ed. (1991)
251. Fosset B., Amatore C.A., Bartelt J.E., Michael A.C., Wightman R.M., Anal. Chem. **63**, 306–314 (1991)
252. Fosset B., Amatore C.A., Bartelt J.E., Wightman R.M., Anal. Chem. **63**, 1403–1408 (1991)
253. Fourier F., *Théorie Analytique de la Chaleur*, vol. I, Didot, Pere et Fils, Paris (1822)
254. Fox L., in: *Numerical solution of Ordinary and Partial Differential Equation* (Edited by L. Fox), Pergamon Press, Oxford, 230–241 (1962)
255. Friedrichs M.S., Friesner R.A., Bard A.J., J. Electroanal. Chem. **258**, 243–264 (1989)
256. Frumkin A., Nekrasov L., Levich B., Ivanov Y., J. Electroanal. Chem. **1**, 84–90 (1959/60)
257. Galus Z., *Fundamentals of Electrochemical Analysis*, Ellis Horwood, New York, 2 ed. (1994), transl. Eds. R.A. Chalmers & W.A.J. Bryce
258. García-Hernández M.T., Castilla J., González-Fernández C.F., Horno J., J. Electroanal. Chem. **424**, 207–212 (1997)
259. Gavaghan D.J., J. Electroanal. Chem. **420**, 147–158 (1997)
260. Gavaghan D.J., J. Electroanal. Chem. **456**, 1–12 (1998)
261. Gavaghan D.J., J. Electroanal. Chem. **456**, 13–23 (1998)
262. Gavaghan D.J., J. Electroanal. Chem. **456**, 25–35 (1998)
263. Gear C.W., in: *Information Processing 68* (Edited by A.J.H. Morrel), North-Holland Publishing Company, Amsterdam, 187–193 (1969)
264. Gear C.W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, New Jersey. (1971)
265. Georganopoulou D.G., Caruana D.J., Strutwolf J., Williams D.E., Faraday Disc. **116**, 109–118 (2000)
266. Gerald C.F., *Applied Numerical Analysis*, Addison–Wesley, Reading, MA, USA, 2nd ed. (1978)
267. Gerischer H., Mattes I., Braun R., J. Electroanal. Chem. **10**, 553–567 (1965)
268. Gilb T., *Software Metrics*, Studentlitteratur, Lund (1976)
269. Glauert M.B., J. Fluid Mech. **1**, 625–643 (1956)
270. Goldberg I.B., Bard A.J., J. Electroanal. Chem. **38**, 313–322 (1972)
271. González C.F., García-Hernández M.T., Horno J., Coll. Czech. Chem. Commun. **57**, 1373–1380 (1992)
272. Gordon P., J. Soc. Indust. Appl. Math **13**, 667–678 (1965)
273. Gosser D.K., Rieger P.H., Anal. Chem. **60**, 1159–1167 (1988)
274. Gosser Jr. D.K., *Cyclic Voltammetry*, VCH, New York and Weinheim, Germany (1993)
275. Gourlay A.R., J. Inst. Maths. Appl. **6**, 375–390 (1970)
276. Gourlay A.R., McGuire G.R., J. Inst. Maths Appl. **7**, 216–227 (1971)
277. Gourlay A.R., Morris J.L., SIAM J. Numer. Anal. **17**, 641–655 (1980)
278. Gourlay A.R., Morris J.L., IMA J. Num. Anal. **1**, 347–357 (1981)
279. Gray D.G., Harrison J.A., J. Electroanal. Chem. **24**, 187–194 (1970)
280. Greenfield P.F., Simulation **22**, 152–154 (1974)
281. Gregory D.P., Riddiford A.C., J. Chem. Soc. 3756–3764 (1956)
282. Gresho P.M., Lee R.L., Computers Fluids **9**, 223–253 (1981)
283. Großmann C., Roos H.G., *Numerik partieller Differentialgleichungen*, Teubner, Stuttgart (1994)

284. Hairer E., Nørsett S.P., Wanner G., *Solving Ordinary Differential Equations I. Nonstiff Problems*, Springer Verlag, Berlin (1987)
285. Hairer E., Wanner G., SIAM J. Num. Anal. **20**, 1206–1209 (1983)
286. Hairer E., Wanner G., *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer Verlag, Berlin (1991)
287. Harriman K., Gavaghan D.J., Houston P., Kay D., Süli E., Electrochem. Commun. **2**, 576–585 (2000)
288. Harriman K., Gavaghan D.J., Houston P., Süli E., Electrochem. Commun. **2**, 567–575 (2000)
289. Harriman K., Gavaghan D.J., Houston P., Süli E., Electrochem. Commun. **2**, 150–156 (2000)
290. Harriman K., Gavaghan D.J., Houston P., Süli E., Electrochem. Commun. **2**, 163–170 (2000)
291. Harriman K., Gavaghan D.J., Houston P., Süli E., Electrochem. Commun. **2**, 157–162 (2000)
292. Harriman K., Gavaghan D.J., Süli E., Electrochem. Commun. **5**, 519–529 (2003)
293. Harriman K., Gavaghan D.J., Süli E., J. Electroanal. Chem. **569**, 35–46 (2004)
294. Harrington M.S., Anderson L.B., Anal. Chem. **62**, 546–550 (1990)
295. Hartree D.R., *Numerical Analysis*, Oxford University Press, Oxford (1958)
296. Hartree D.R., Womersley J.R., Proc. Roy. Soc. London Ser. A **161**, 353–367 (1937)
297. Heinze J., J. Electroanal. Chem. **124**, 73–86 (1981)
298. Heinze J., Ber. Bunsenges. Phys. Chem. **85**, 1096–1103 (1984)
299. Heinze J., Angew. Chem. **105**, 1327–1349 (1993)
300. Heinze J., Störzbach M., Ber. Bunsenges. Phys. Chem. **90**, 1043–1048 (1986)
301. Heinze J., Störzbach M., Mortensen J., J. Electroanal. Chem. **165**, 61–70 (1984)
302. Henrici P., *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley, New York (1962)
303. Hertl P., Speiser B., J. Electroanal. Chem. **217**, 225–238 (1987)
304. Hindmarsh A.C., Petzold L.R., Computers in Physics **9**, 148–155 (1995)
305. Horno J., García-Hernández, González-Fernández, J. Electroanal. Chem. **377**, 53–60 (1994)
306. Horno J., García-Hernández M.T., J. Electroanal. Chem. **352**, 83–97 (1993)
307. Horno J., García-Hernández M.T., Castilla J., González-Fernández C.F., Electroanalysis **8**, 1145–1149 (1996)
308. Horno J., González C.F., Hayas A., J. Comp. Phys. **118**, 310–319 (1995)
309. Horno J., González-Fernández C.F., Hayas A., González-Caballero F., Biophys. J. **55**, 527–535 (1989)
310. Hsu C.T., Shao M.J., Lin S.Y., Langmuir **16**, 3187–3194 (2000)
311. Huber A., Monatsh. Math. Phys. **47**, 240–246 (1939)
312. Hunter I.C., Jones I.P., *Numerical experiments on the effects of strong grid stretching in finite difference calculations*, Tech. Rep. AERE R-10301, United Kingdom Atomic Energy Authority, Harwell (1981)
313. Imbeaux J.C., Savéant J.M., J. Electroanal. Chem. **28**, 325–338 (1970)
314. Jain M.K., *Numerical Solution of Differential Equations*, Wiley Eastern, New Delhi, 2 ed. (1984)
315. Jehring H., *Elektrosorptionsanalyse mit der Wechselstrompolarographie*, Akademie-Verlag, Berlin (1974)

316. Jin B., Qian W., Zhang Z., Shi H., J. Electroanal. Chem. **411**, 29–36 (1996)
317. Jin B., Qian W., Zhang Z., Shi H., J. Electroanal. Chem. **417**, 45–51 (1996)
318. Jin B., Qian W., Zhang Z., Shi H., J. Electroanal. Chem. **411**, 19–27 (1996)
319. Jin B.K., Shi H.S., Zhang Z.X., Chem. J. Chin. Univ. **17**, 1052–1055 (1996), [in Chinese, Eng. abstract]
320. Johannsen K., Britz D., Comput. Chem. **23**, 33–41 (1999)
321. Joslin T., Pletcher D., J. Electroanal. Chem. **49**, 171–186 (1974)
322. Josserand J., Morandini J., Lee H.J., Ferrigno R., Girault H.H., J. Electroanal. Chem. **468**, 42–52 (1999)
323. Juozėnas A., Šidlauskas V., Jurevičius D., Chemija (1), 13–17 (1993)
324. Jurczakowski R., Orlik M., J. Electroanal. Chem. **478**, 118–127 (1999)
325. Kader B.A., Sov. Electrochem. **13**, 417–423 (1977)
326. Kakihana M., Ikeuchi H., Satô G.P., Tokuda K., J. Electroanal. Chem. **108**, 381–383 (1980)
327. Kakihana M., Ikeuchi H., Satô G.P., Tokuda K., J. Electroanal. Chem. **117**, 201–211 (1981)
328. Kálnay de Rivas E., J. Comp. Phys. **10**, 202–210 (1972)
329. Kantorovich L.V., Doklady Akad. Nauk. **2**, 532–536 (1934), in Russian, with a French translation added
330. Kantorowitsch L.W., Krylow W.I., *Näherungsmethoden der höheren Analyse*, VEB Deutscher Verlag der Wissenschaften, Berlin (1956)
331. Karaoglanoff Z., Z. Elektrochem. **12**, 5–16 (1906)
332. Kavanaugh T.C., Friedrichs M.S., Friesner R.A., Bard A.J., J. Electroanal. Chem. **283**, 1–14 (1990)
333. Kay J.M., Nedderman R.M., *An Introduction to Fluid Mechanics and Heat Transfer*, Cambridge University Press, Cambridge, UK, 3rd ed. (1974)
334. Keast P., Mitchell A.R., Computer J. **9**, 110–114 (1966)
335. Kermiche-Aouanouk F., Daguenet M., J. Chim. Phys. Physicochim. Biol. **69**, 1705–1710 (1972), in French
336. Ketter J.K., Forry S.P., Wightman R.M., Feldberg S.W., Electrochem. Solid-State Lett. **7**, E18–E22 (2004)
337. Khaliq A.Q.M., Wade B.A., J. Comput. Meth. Sci. Eng. **1**, 107–124 (2001)
338. Kimble M.C., White R.E., Comput. Chem. Eng. **14**, 921–924 (1990)
339. Klinger J., Conway B.E., Angerstein-Kozlowska H., Comput. Chem. **2**, 117–129 (1978)
340. Klymenko O.V., Evans R.G., Hardacre C., Svir I.B., Compton R.G., J. Electroanal. Chem. **571**, 211–221 (2004)
341. Kopal Z., *Numerical Analysis*, Chapman & Hall, London (1955)
342. Koryta J., Coll. Czech. Chem. Commun. **18**, 206–213 (1953)
343. Laasonen P., Acta Math. **81**, 309–317 (1949)
344. Lambert J.D., *Computational Methods in Ordinary Differential Equations*, Wiley, New York (1972)
345. Lang J., Appl. Num. Math. **18**, 223–240 (1995)
346. Lang J., Chem. Eng. Sci. **51**, 1055–1070 (1996)
347. Lang J., *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems*, Springer, Berlin (2001)
348. Lang J., Verwer J., BIT **41**, 731–738 (2001)
349. Lantelme F., Groult H.H., Kumagai N., Electrochim. Acta **45**, 3171–3180 (2000)

350. Lapidus L., Pinder G.F., *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley, New York (1982)
351. Lapidus L., Seinfeld J.H., *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York (1971)
352. Larkin B.K., Math. Comp. **18**, 196–202 (1964)
353. Lasia A., Grégoire D., J. Electrochem. Soc. **142**, 3393–3399 (1995)
354. Lavagnini I., Pastore P., Magno F., J. Electroanal. Chem. **333**, 1–10 (1992)
355. Lavagnini I., Pastore P., Magno F., Amatore C.A., J. Electroanal. Chem. **316**, 37–47 (1991)
356. Lawson J.D., Morris J.L., SIAM J. Numer. Anal. **15**, 1212–1224 (1978)
357. Lemos M.A., Port. Electrochim. Acta **15**, 163–187 (1997)
358. Lemos M.A.N.D.A., Lemos F., Papadopoulos N., Pombeiro A.J.L., Port. Electrochim. Acta **16**, 175–180 (1998)
359. Lemos M.A.N.D.A., Pombeiro A.J.L., Port. Electrochim. Acta **10**, 89–99 (1992)
360. Lemos M.A.N.D.A., Pombeiro A.J.L., in: *Molecular Electrochemistry of Inorganic, Bioinorganic and Organometallic Compounds* (Edited by A.J.L. Pombeiro, J.A. McCleary), Kluwer Academic Publ., Netherlands, 477–482 (1993)
361. Lerke S.A., Evans D.H., Feldberg S.W., J. Electroanal. Chem. **296**, 299–315 (1990)
362. Levich V.G., *Physicochemical Hydrodynamics*, Prentice-Hall, New Jersey (1962)
363. Licht S., Cammarata V., Wrighton M.S., J. Phys. Chem. **94**, 6133–6140 (1990)
364. Lindberg B., BIT **11**, 29–52 (1971)
365. Lingane P.J., Anal. Chem. **36**, 1723–1726 (1964)
366. Liskovets O.A., Diff. Urav. **1**, 1662–1677 (1965)
367. Liu S.L., Chem. Eng. Sci. **22**, 871–881 (1967)
368. Liu S.L., AIChE J. **15**, 334–338 (1969)
369. Lopéz-García J.J., Grosse C., Horno J., J. Colloid. Interf. Sci. **254**, 287–295 (2002)
370. Lorenz W., Z. Elektrochem. **62**, 192–200 (1958)
371. Lovrić M., Kormorsky-Lovrić Ŝ., Bull. Soc. Chim. Beograd **46**, 93–98 (1981)
372. Ludwig K., Rajendran L., Speiser B., J. Electroanal. Chem. **568**, 203–214 (2004), see Erratum, ibid. 571 (2004) 119
373. Ludwig K., Rajendran L., Speiser B., J. Electroanal. Chem. **571**, 119 (2004)
374. Luskin M., Rannacher R., Appl. Anal. **14**, 117–135 (1982)
375. Macpherson J.V., Unwin P.R., J. Phys. Chem. **100**, 19475–19483 (1996)
376. Magno F., Bontempelli G., Perosa D., Anal. Chim. Acta **147**, 65–76 (1983)
377. Mahon P.J., Oldham K.B., Electrochim. Acta **49**, 5041–5048 (2004)
378. Mandin P., Pauporte T., Fanouillère P., Lincot D., J. Electroanal. Chem. **566**, 159–173 (2004)
379. Margarit J., Dabosi G., Lévy M., Bull. Soc. Chim. Fr. 1509–1512 (1975)
380. Margarit J., Lévy M., J. Electroanal. Chem. **49**, 369–376 (1974)
381. Marques da Silva B., Avaca L.A., Gonzalez E.R., J. Electroanal. Chem. **250**, 457–460 (1988)
382. Marques da Silva B., Avaca L.A., Gonzalez E.R., J. Electroanal. Chem. **269**, 1–14 (1989)
383. Martin R.D., Unwin P.R., J. Electroanal. Chem. **439**, 123–136 (1997)

384. Martin R.D., Unwin P.R., Anal. Chem. **70**, 276–284 (1998)
385. Martínez-Ortiz F., Private communication to D. Britz (2004)
386. Mastragostino M., Nadjo L., Saveant J.M., Electrochim. Acta **13**, 721–749 (1968)
387. Matsuda H., J. Electroanal. Chem. **15**, 325–336 (1967)
388. Matsuda H., J. Electroanal. Chem. **16**, 153–164 (1968)
389. Matsuda H., Ayabe Y., Z. Elektrochem **59**, 494–503 (1955)
390. Mauzeroll J., Hueske E.A., Bard A.J., Anal. Chem. **75**, 3880–3889 (2003)
391. McCormick S., Nature **337**, 205 (1989)
392. Metcalf M., Reid J., *The F Programming Language*, Oxford University Press, Oxford, UK (1996)
393. Metcalf M., Reid J., *Fortran 90/95 Explained*, Oxford University Press, Oxford, UK (1996)
394. Michael A.C., Wightman R.M., Amatore C.A., J. Electroanal. Chem. **267**, 33–45 (1989)
395. Miller R., Colloid & Polymer Sci. **259**, 375–381 (1981)
396. Miller R., Kretzschmar G., Colloid & Polymer Sci. **258**, 85–87 (1980)
397. Miller R., Lunkenheimer K., Z. phys. Chem. (Leipzig) **259**, 863–868 (1978)
398. Miller R., Lunkenheimer K., Kretzschmar G., Colloid & Polymer Sci. **257**, 1118–1120 (1979)
399. Mizushina T., Adv. Heat Mass Trans. **7**, 87–161 (1971)
400. Mocak J., Electrochem. Commun. **4**, 803–807 (2002)
401. Mocak J., Bond A., J. Electroanal. Chem. **561**, 191–202 (2004)
402. Mocak J., Feldberg S.W., J. Electroanal. Chem. **378**, 31–37 (1994)
403. Mohilner D.M., in: *Electroanal. Chem.* (Edited by A.J. Bard), Marcel Dekker, New York, vol. 1, 241–409 (1966)
404. Morse P.M., Feshbach H., *Methods in Theoretical Physics*, McGraw-Hill, NY (1953)
405. Moya A.A., Hayas A., Horno J., Ber. Bunsenges. phys. Chem. **99**, 1037–1042 (1995)
406. Nagel L.W., *SPICE (simulation Program with Integrated Circuit Emphasis)*, Tech. Rep. ERL-m382-1977, Electronics Research laboratory, University of California, Berkeley (1977)
407. Nann T., *Digitale Simulation in der Elektrochemie mit der Methode der Finiten Elementen*, Ph.D. thesis, Albert-Ludwigs-Universität zu Freiburg im Breisgau (1997), publ. by Shaker Verlag, Aachen
408. Nann T., Heinze J., Electrochem. Commun. **1**, 289–294 (1999)
409. Nann T., Heinze J., Electrochimica Acta **48**, 3975–3880 (2003)
410. Nernst W., Z. phys. Chem. **47**, 52–55 (1904)
411. Newman J., J. Electrochem. Soc. **113**, 501–502 (1966)
412. Newman J., Ind. Eng. Chem. Fundam. **7**, 514–517 (1968)
413. Newman J., *Electrochemical Systems*, Prentice-Hall, New Jersey (1973)
414. Nguyen T.V., White R., Comput. Chem. Eng. **11**, 543–546 (1987)
415. Nicholson R.S., Anal. Chem. **37**, 667–671 (1965)
416. Nicholson R.S., Olmstead M.L., in: *Comput. Chem. Instrum.* (Edited by J. Mattson, H.B. Mark Jr, H.C. MacDonald Jr), Marcel Dekker, New York, vol. 2, 119–139 (1972)
417. Nicholson R.S., Shain I., Anal. Chem. **36**, 706–723 (1964)

418. Nielsen M.F., Almdal K., Hammerich O., Parker V.D., Act Chem. Scand. A **41**, 423–440 (1987)
419. Nikolić S., *Digitalna simulacija elektrodnih reakcija za pulsnu polarografiju i srodne tehnike*, Master's thesis, Zagreb University (1983)
420. Nolan J.E., Plambeck J.A., J. Electroanal. Chem. **294**, 1–20 (1990)
421. Noumerov B.V., Monthly Not. Roy. Astr. Soc. **84**, 592–601 (1924)
422. Noye J., in: *Numerical solutions of PDE's* (Edited by J. Noye), Queen's College, Melbourne (1982), 3–137
423. Noye J., in: *Computational techniques for differential equations* (Edited by J. Noye), Elsevier, Amsterdam, 95–354 (1984)
424. O'Brien G.G., Hyman M.A., Kaplan S., J. Math. Phys. **29**, 223–251 (1950)
425. Oldham K., J. Electroanal. Chem. **122**, 1–17 (1981)
426. Oldham K.B., J. Electroanal. Chem. **105**, 373–375 (1979)
427. Oldham K.B., Zoski C.G., J. Electroanal. Chem. **256**, 11–19 (1988)
428. Oleinick A., Amatore C., Svir I., Electrochem. Commun. **6**, 588–594 (2004)
429. Orlik M., J. Electroanal. Chem. **434**, 139–152 (1997)
430. Østerby O., *The error of the Crank-Nicolson method for linear parabolic equations with a derivative boundary condition*, Report PB-534, DAIMI, Aarhus University (1998)
431. Østerby O., *Five ways of reducing the Crank-Nicolson oscillations*, Tech. Rep. Daimi PB-558, Dept. of Computer Science, Aarhus University (2002)
432. Østerby O., BIT **43**, 811–822 (2003)
433. Pao Y.H., Daugherty R.J., *Time-dependent viscous incompressible flow past a finite flat plate*, Tech. Rep. Rept. DI-82-0822, Boeing Sci. Res. Lab. (1969)
434. Pastore P., Magno F., Lavagnini I., Amatore C., J. Electroanal. Chem. **301**, 1–13 (1991)
435. Patankar S.V., *Numerical Heat Transfer and Fluid Flow*, Hemiphere Publishing Corp., New York (1980)
436. Peaceman D.W., Rachford H.H., J. Soc. Ind. Appl. Math. **3**, 28–41 (1955)
437. Pearson C.E., Math. Comput. **19**, 570–576 (1965)
438. Pedersen S.U., Christensen T.B., Thomasen T., Daasbjerg K., J. Electroanal. Chem. **454**, 123–143 (1998)
439. Penar J., Ann. Univ. Marie Curie-Skłodowska Lublin-Polonia **46/47**, 119–172 (1991/2), in Polish
440. Penar J., Persona A., Stawinski A., Pol. J. Chem. **67**, 529–540 (1993)
441. Petzold L., in: *Scientific Computing* (Edited by R.S. Stepleman, M. Carver, R. Peskin, W.F. Ames, R. Vichnevetsky), North Holland Publ. Co., Amsterdam (1983), vol. 1, IMACS Trans. Sci. Comp., 10th IMACS World Congress on Systems Simulation and Scientific Computation, Montreal, Canada, August 1982, 65–68
442. Phillips C.G., Jansons K.M., Proc. Roy. Soc. Lond. A **428**, 431–439 (1990)
443. Pons B.S., Schmidt P.P., Electrochim. Acta **25**, 987–993 (1980)
444. Pons B.S., Speiser B., McAleer J.F., Electrochim. Acta **27**, 1177–1179 (1982)
445. Pons B.S., Speiser B., McAleer J.F., Schmidt P.P., Electrochim. Acta **27**, 1711–1714 (1982)
446. Pons S., in: *Electroanal. Chem.* (Edited by A.J. Bard), Marcel Dekker, New York, vol. 13, 115–190 (1984)
447. Postlethwaite T.A., Hutchinson J.E., Murray R., Fosset B., Amatore C., Anal. Chem. **68**, 2951–2958 (1996)

448. Prandtl L., Tietjens O.G., *Applied Hydro- and Aerodynamics*, Dover, New York (1934)
449. Prater K.B., Chem. Instrum. **3**, 259–269 (1972)
450. Prater K.B., Bard A.J., J. Electrochem. Soc. **117**, 207–213 (1970)
451. Prater K.B., Bard A.J., J. Electrochem. Soc. **117**, 335–340 (1970)
452. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P., *Numerical Recipes in Fortran*, Cambridge University Press, Cambridge, 2 ed. (1986)
453. Pritzker M.D., J. Electroanal. Chem. **243**, 57–80 (1988)
454. Qian W., Jin B., Diao G., Zhang Z., Shi H., J. Electranal. Chem. **414**, 1–10 (1996)
455. Qian W., Jin B., Shi H., Zhang Z., J. Electroanal. Chem. **439**, 29–36 (1997)
456. Qian W., Jin B.K., Shi H.S., Yu J.S., Zhang Z.X., Acta. Chim. Sinica **55**, 1108–1115 (1997)
457. Qiu F.L., Fisher A.C., Henley I.E., Dryfe R.A.W., Electrochem. Commun. **5**, 169–174 (2003)
458. Rajantie H., Strutwolf J., Williams D.E., J. Electroanal. Chem. **500**, 108–120 (2001)
459. Rampazzo L., Electrochim. Acta **14**, 733–739 (1969)
460. Randles J.E.B., Trans. Faraday Soc. **44**, 327–338 (1948)
461. Rannacher R., Z. Angew. Math. Mech. **62**, T346–T348 (1982)
462. Rannacher R., Numer. Math. **43**, 309–327 (1984)
463. Rees N.V., Klymenko O.V., Maisonhaute E., Coles B.A., Compton R.G., J. Electroanal. Chem. **542**, 23–32 (2003)
464. Reinert K.E., Berg H., Monatsber. Deut. Akad. Wiss. Berlin **4**, 26–32 (1962)
465. Reinmuth W.H., J. Phys. Chem. **65**, 473–476 (1961)
466. Ribeiro L.M.D., Lemos M.A.N.D.A., Pombeiro A.J.L., Sobota P., Russ. J. Electrochem. **31**, 1009–1015 (1995)
467. Rice J.R., *Numerical Methods, Software, and Analysis*, McGraw-Hill International, Auckland (1983)
468. Richardson L.F., Phil. Trans. A **210**, 307–357 (1911)
469. Richardson L.F., Phil. Trans. A **226**, 299–349 (1927)
470. Richtmyer R.D., *Difference Methods for Initial-Value Problems*, Interscience, New York (1957)
471. Richtmyer R.D., Morton K.W., *Difference Methods for Initial-Value Problems*, Interscience, New York (1967)
472. Rizzo F.J., Shippy D.J., AIAA J. **8**, 2004–2009 (1970)
473. Roche M., Numer. Math. **52**, 45–63 (1988)
474. Rosenbrock H., Computer J. **5**, 329–300 (1962/3)
475. Rothe E., Math. Ann. **102**, 651–670 (1930)
476. Rudolph M., J. Electroanal. Chem. **314**, 13–22 (1991)
477. Rudolph M., in: *Physical Electrochemistry* (Edited by I. Rubinstein), Marcel Dekker, New York, 81–129 (1995)
478. Rudolph M., J. Electroanal. Chem. **529**, 97–108 (2002)
479. Rudolph M., J. Electroanal. Chem. **543**, 23–39 (2003)
480. Rudolph M., J. Electroanal. Chem. **558**, 171–176 (2003)
481. Rudolph M., J. Electroanal. Chem. **571**, 289–307 (2004)
482. Rudolph M., Reddy D.P., Feldberg S.W., Anal. Chem. **66**, 589A–600A (1994)
483. Ružić I., J. Electroanal. Chem. **144**, 433–436 (1983)
484. Ružić I., J. Electroanal. Chem. **199**, 431–435 (1986)

485. Ružić I., Britz D., Acta Chem. Scand. **45**, 1087–1089 (1991)
486. Ružić I., Feldberg S., J. Electroanal. Chem. **50**, 153–162 (1974)
487. Ružić I., Smith D.E., J. Electroanal. Chem. **57**, 129–139 (1974)
488. Safford L.K., Weaver M.J., J. Electroanal. Chem. **261**, 241–247 (1989)
489. Safford L.K., Weaver M.J., J. Electroanal. Chem. **312**, 69–96 (1991)
490. Saito Y., Rev. Polarog. (Japan) **15**, 177–187 (1968)
491. Salaun P., Josserand J., Morandini J., Girault H.H., Buffle J., J. Electroanal. Chem. **566**, 147–158 (2004)
492. Sanchez G., Codina G., Aldaz A., J. Chem. Educ. **68**, 489–490 (1991)
493. Sand H.J.S., Z. phys. Chem. **35**, 641–651 (1900)
494. Sandifer J.R., Buck R.P., J. Electroanal. Chem. **49**, 161–170 (1974)
495. Sanz-Serna J.M., Christie I., J. Comput. Phys. **67**, 348–360 (1986)
496. Saul'yev V.K., *Integration of Equations of Parabolic Type by the Method of Nets*, Pergamon Press, New York (1964)
497. Schiesser W.E., *The Numerical Method of Lines Integration of Partial Differential Equations*, Academic Press, San Diego (1991)
498. Schlichting H., *Grenzschicht-Theorie*, G. Braun, Karlsruhe (1965)
499. Schmidt F.W., Zeldin B., AIChE J. **15**, 612–614 (1969)
500. Seddon B.J., Girault H.H., Eddowes M.J., J. Electroanal. Chem. **266**, 227–238 (1989)
501. Seeber R., Stefani S., Anal. Chem. **53**, 1011–1016 (1981)
502. Selzer Y., Mandler D., Electrochem. Commun. **1**, 569–575 (1999)
503. Selzer Y., Mandler D., Anal. Chem. **72**, 2383–2390 (2000)
504. Serna C., López-Tenés M., González J., Electrochim. Acta **46**, 2699–2709 (2001)
505. Ševčík A., Coll. Czech. Chem. Commun. **13**, 349–377 (1948)
506. Shih T.M., *Numerical Heat Transfer*, Hemisphere Publ. Corp., Washington (1984)
507. Shoup D., Szabo A., J. Electroanal. Chem. **140**, 237–245 (1982)
508. Shoup D., Szabo A., J. Electroanal. Chem. **160**, 19–26 (1984)
509. Shoup D., Szabo A., J. Electroanal. Chem. **160**, 1–17 (1984)
510. Shoup D., Szabo A., J. Electroanal. Chem. **160**, 27–31 (1984)
511. Shoup D., Szabo A., J. Electroanal. Chem. **199**, 437–441 (1986)
512. Slevin C.J., Macpherson J.V., Unwin P.R., J. Phys. Chem. B **101**, 10851–10859 (1997)
513. Smith C.P., White H.S., Anal. Chem. **65**, 3343–3353 (1993)
514. Smith G.D., *Numerical Solution of Partial Differential Equations*, Oxford University Press, Oxford, 3 ed. (1985)
515. Soos Z.G., Lingane P.J., J. Phys. Chem. **68**, 3821–3828 (1964)
516. Sparrow E.M., Gregg J.L., J. Heat Trans. **81C**, 249–252 (1959)
517. Speiser B., in: *Softwareentwicklung in der Chemie 3* (Edited by G. Gauglitz), Springer, Berlin, 321–332 (1989)
518. Speiser B., Comput. Chem. **14**, 127–140 (1990)
519. Speiser B., J. Electroanal. Chem. **301**, 15–35 (1991)
520. Speiser B., Anal. Chim. Acta **243**, 301–310 (1991)
521. Speiser B., Acta Chem. Scand. **47**, 1238–1240 (1993)
522. Speiser B., J. Electroanal. Chem. **374**, 280–282 (1994)
523. Speiser B., in: *Electroanal. Chem.* (Edited by A.J. Bard, I. Rubinstein), Marcel Dekker, New York, vol. 19, 1–108 (1996)

524. Spiegel M.R., *Advanced Calculus*, McGraw-Hill, New York (1963)
525. Stanley R.P., *Enumerative Combinatorics*, vol. 1, Cambridge University Press, Cambridge, New York (1997)
526. Steffen B., Rousar I., Electrochim. Acta **40**, 379–386 (1995)
527. Stone H.L., SIAM J. Num. Anal. **5**, 530–558 (1968)
528. Strikwerda J.C., *Finite Difference Schemes and Partial Differential Equations*, Wadsworth and Brooks/Cole, Pacific Grove, California (1989)
529. Strutwolf J., *Digitale Simulation elektrochemischer Systeme: Untersuchungen zeitabhängiger Phänomene an rotierenden Scheibenelektroden und Analyse von Cyclovoltammogrammen durch direkte Simulation*, Ph.D. thesis, Universität Bielefeld, Germany (1995)
530. Strutwolf J., Barker A.L., Gonsalves M., Caruana D.J., Unwin P.R., Williams D.E., Webster J.P.R., J. Electroanal. Chem. **483**, 163–173 (2000)
531. Strutwolf J., Britz D., Comput. Chem. **25**, 205–214 (2001)
532. Strutwolf J., Britz D., J. Electroanal. Chem. **566**, 15–23 (2004)
533. Strutwolf J., Schoeller W.W., Electroanalysis **8**, 1034–1039 (1996)
534. Strutwolf J., Schoeller W.W., Electroanalysis **9**, 1403–1408 (1997)
535. Strutwolf J., Williams D.E., Electroanalysis **11**, 487–493 (1999)
536. Strutwolf J., Williams D.E., Electroanalysis **submitted**, 0–0 (2004)
537. Sundqvist H., Veronis G., Tellus **22**, 26–31 (1970)
538. Svir I.B., Golovenko V.M., Electrochem. Commun. **3**, 11–15 (2001)
539. Svir I.B., Klimenko A.V., Compton R.G., Radiotekh. **118**, 92–101 (2001)
540. Svir I.B., Oleinick A.I., J. Electroanal. Chem. **499**, 30–38 (2001)
541. Svir I.B., Oleinick A.I., Compton R.G., J. Electroanal. Chem. **560**, 117–126 (2003)
542. Svir I.B., Oleinick A.I., Klimenko A.V., J. Electroanal. Chem. **513**, 119–125 (2001)
543. Szabo A., Cope D.K., Tallman D.E., Kovach P.M., Wightman R.M., J. Electroanal. Chem. **217**, 417–423 (1987)
544. Szymanska J., Palys M.J., Van den Bossche B., Anal. Chem. **76**, 5937–5944 (2004)
545. Taylor G., Girault H.H., McAleer J., J. Electroanal. Chem. **293**, 19–44 (1990)
546. Thomas L.H., *Elliptic Problems in Linear Difference Equations over a Network*, Watson Scientific Computing Laboratory, Columbia University, New York (1949)
547. Thompson J.F., Appl. Num. Math. **1**, 3–27 (1985)
548. Treanor C.E., Math. Comp. **20**, 39–45 (1966)
549. Twinanga E.W., *A Guide to Circuit Simulation and Analysis Using PSPICE*, Prentice-Hall, New Jersey (1992)
550. Urban P., Speiser B., J. Electroanal. Chem. **241**, 17–31 (1988)
551. Van Den Bossche B., Bortels L., Deconinck J., Vandeputte S., Hubin A., J. Electroanal. Chem. **397**, 35–44 (1995)
552. Van Den Bossche B., Floridor G., Deconinck J., Van Den Winkel P., Hubin A., J. Electroanal. Chem. **531**, 61–70 (2002)
553. van den Houwen P.J., Sommeijer B.P., Appl. Num. Math. **11**, 169–188 (1993)
554. van Leeuwen H.P., de Jong H.G., Holub K., J. Electroanal. Chem. **260**, 213–220 (1989)
555. Varadi M., Pungor E., Anal. Chim. Acta **80**, 31–37 (1975)
556. Varco Shea T., Bard A.J., Anal. Chem. **59**, 2101–2111 (1987)

557. Verbrugge M.W., Baker D.R., J. Phys. Chem. **96**, 4572–4580 (1992)
558. Verwer J.G., Sanz-Serna J.M., Computing **33**, 297–313 (1984)
559. Vetter K., *Elektrochemische Kinetik*, Springer, Berlin (1961)
560. Vielstich W., Z. Elektrochem. **57**, 646–655 (1953)
561. Villa C.M., Chapman T.W., Ind. Eng. Chem. Res. **34**, 3445–3453 (1995)
562. Villadsen J., Michelsen M.L., *Solution of differential equation models by polynomial approximation*, Prentice-Hall, New Jersey (1978)
563. Villadsen J.V., Stewart W.E., Chem. Eng. Sci. **22**, 1483–1501 (1967)
564. von Kármán T., Z. Angew. Math. Mech. **1**, 233–252 (1921)
565. von Neumann J., Richtmyer R.D., J. Appl. Phys. **21**, 232–237 (1950)
566. Voss R.F., Tomkiewicz M., J. Electrochem. Soc. **132**, 371–375 (1985)
567. Weinberg G.M., *The Psychology of Computer Programming*, Van Nostrand Reinhold, New York (1971)
568. Welford P.J., Brookes B.A., Climent V., Compton R.G., J. Electroanal. Chem. **513**, 8–15 (2001)
569. Wesseling P., *An Introduction to Multigrid Methods*, Wiley, New York (1992)
570. White R.E., Ind. Eng. Chem. Fundam. **17**, 367–369 (1978)
571. Whiting L.F., Carr P.W., J. Electroanal. Chem. **81**, 1–20 (1977)
572. Wiesner K., Chem. Listy **41**, 6–8 (1947)
573. Williams D.E., MacPherson J., in: *Comprehensive Chemical Kinetics* (Edited by R.G. Compton, G. Hancock), Elsevier, Amsterdam, vol. 37, 369–438 (1999)
574. Wipf D.O., Wightman R.M., Anal. Chem. **60**, 2460–2464 (1988)
575. Wood W.L., Lewis R.W., Int. J. Num. Methods Eng. **9**, 679–689 (1975)
576. Wright K., Computer J. **6**, 358–365 (1964)
577. Wu B., White R.E., Comput. Chem. Eng. **28**, 303–309 (2004)
578. Wu Y., Wang Z., Electrochim. Acta **44**, 2281–2286 (1999)
579. Yamada J., Matsuda H., J. Electroanal. Chem. **44**, 189–198 (1973)
580. Yen S.C., Chapman T.W., J. Electroanal. Chem. **135**, 305–312 (1982)
581. Zeiri L., Younes O., Efrima S., Deutsch M., J. Phys. Chem. B **101**, 9299–9308 (1997)
582. Zhang W.S., Zhang X.W., J. Electroanal. Chem. **445**, 55–62 (1998)
583. Zhang Y., Cheh H.Y., J. Electrochem. Soc. **146**, 850–856 (1999)
584. Zlatev Z., Wasniewski J., Schaumburg K., in: *Lect. Notes Comp. Sci.* (Edited by G. Goos, J. Hartmanis), Springer, Berlin, vol. 121 (1981)
585. Zoski C.G., Mirkin M.V., Anal. Chem. **74**, 1986–1992 (2002)

Index