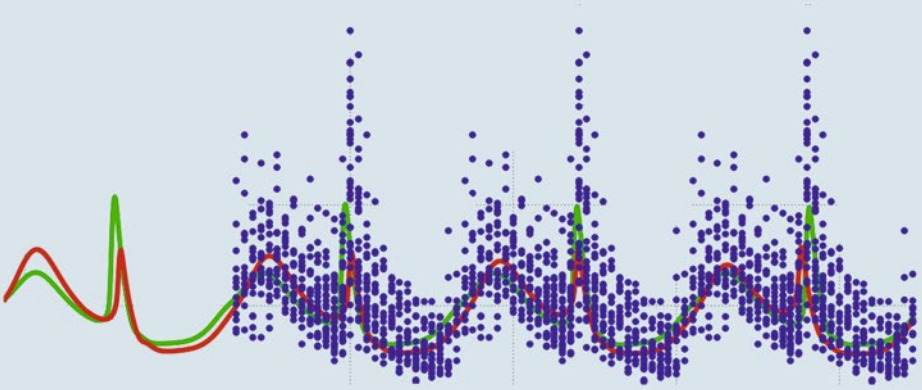


TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING 12

Peter Deuflhard · Susanna Röblitz

A Guide to Numerical Modelling in Systems Biology

Springer

Texts in Computational Science and Engineering 12

Editors

More information about this series at
http://www.springer.com/series/5151

Peter Deuflhard • Susanna Röblitz

A Guide to Numerical Modelling in Systems Biology

Springer

Peter Deuflhard
Zuse-Institut Berlin (ZIB)
Berlin, Germany

Susanna Röblitz
Zuse Institute Berlin (ZIB)
Berlin, Germany

ISSN 1611-0994 ISSN 2197-179X (electronic)
Texts in Computational Science and Engineering
ISBN 978-3-319-20058-3 ISBN 978-3-319-20059-0 (eBook)
DOI 10.1007/978-3-319-20059-0

Library of Congress Control Number: 2015944536

Mathematics Subject Classification (2010): 65F20, 65F35, 65L05, 65L06, 65L09, 92-01, 92-08, 92C45, 92C42

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

In recent years, *systems biology* has emerged as an interdisciplinary research field combining biology with computer science and mathematics. Computational models for complex biological processes are the general paradigm, predominantly in terms of ordinary differential equations (ODEs). The main scientific tasks are mathematical modelling of biochemical and physiological processes, numerical simulation of the dynamics of biological networks, and identification of model parameters via a comparison with measurement data.

Typical books on systems biology do not go into the necessary detail as far as these topics are concerned. At best, they merely mention algorithmic approaches, but without offering a deeper understanding. On the other hand, books on ODEs or parameter identification are typically written for a mathematical community, nearly unreadable for systems biologists or, more generally, for computational biologists. As authors of the present book, we have worked hard to fill this gap: Our aim has been to make the important mathematical issues readable and to focus on systems biological needs. What came out is *not a book on systems biology, but on computational methods in systems biology*.

Our book is based on *university courses* repeatedly given to students of bioinformatics who only had moderate knowledge of mathematics. The idea of the courses had been to convey mathematical insight as far as it is indispensable for systems biological modelling. Consequently, the book aims at teaching the necessary mathematical prerequisites by means of many examples rather than by theorems. This does not and cannot mean to avoid mathematical formulas as a whole.

Throughout the text, numerical *software* is discussed in terms of its strengths and weaknesses with respect to various systems biological issues. Web addresses are included where the mentioned software can be downloaded.

Acknowledgements First of all, we want to thank our ZIB colleague Thomas Dierkes for his patience and invaluable help in the preparation of the computational results for the GynCycle model (Sect. 3.5.3). We also wish to thank our ZIB colleagues Rainald Ehrig and Claudia Stötzel for their helpful groundwork. Moreover, we are indebted to Matthias König and Hermann-Georg

Holzhütter from Charité Berlin for their support in the example presented in Sect. 2.5.3 (Warburg effect of tumor cells). Finally, we are grateful to Pooja Gupta for her careful reading of earlier drafts of the manuscript.

Berlin, Germany
May 2015

Peter Deuflhard
Susanna Röblitz

Contents

Outline

This book is divided into the following three chapters:

- Chapter 1: mathematical modelling of biochemical and physiological processes
- Chapter 2: numerical simulation of the dynamics of biological networks
- Chapter 3: identification of model parameters via a comparison with measurement data

A few mathematically more challenging topics are postponed to an Appendix. Throughout the text, numerical software is discussed in terms of its strengths and weaknesses.

Chapter 1 works out the basics of mathematical modelling in systems biology. The first section starts with a short overview about initial value problem types that occur in systems biology – deliberately omitting periodic boundary value problems that do occur quite often but are regarded as technically too complex for the present elementary text. To start with, two simple model problems are worked out in some detail: one from population dynamics and one on multiple dose administration of drugs. Next, the assembly of large ODE networks from simple chemical or physiological mechanisms is described. Reasons are given why the so-called Michaelis-Menten kinetics is no longer needed in the numerical simulation of such systems. For parts of reaction networks, where only the properties "stimulating" or "inhibiting" are known, the formulation in terms of Hill functions is given. Compartment models are introduced by an illustrative example. Finally, necessary mathematical background material is presented as far as it seems important for the class of applications in question. Main topics therein are the uniqueness and sensitivity of solutions as well as asymptotic stability. As the book is mainly addressed to students with only elementary knowledge of mathematics, mathematical contents are typically explained by examples rather than by theorems. Attention deliberately focuses on consequences for practical calculations.

Given possibly large ODE models for systems biological networks, **Chap. 2** deals with their numerical simulation. For this purpose, various numerical integrators for initial value problems are described in necessary detail. To begin with, the focus is on the discrepancy between *controllable local* and *not controllable*

global discretization errors. Stability concepts for discretizations in general lead to an elementary pragmatic understanding of the term "stiffness" of ODE systems. In the remaining chapter, different families of integrators such as one-step methods, extrapolation methods, and multistep methods are characterized. From a practical point of view, they are divided into explicit, implicit, and linearly implicit methods and discussed in terms of their structural strengths and weaknesses. The chapter ends with a general roadmap about numerical integration methods; this roadmap is understood as help in the decision about which integrator to use for which kind of problem. For illustration purposes, two model problems are presented that seem to be quite similar but require different numerical integrators: one of them stiff and the other one non-stiff. Finally, a rather complex problem from tumor cell biology including parameter studies is worked out where computational speed really matters.

Chapter 3 deals with parameter identification, the most important question in systems biology. The whole chapter is devoted to show *why parameter identification problems are not just usual optimization problems*, but are inverse problems with a statistical background. The arising subtle issues are often overlooked in the present systems biological literature. Parameter identification problems in ODE models typically arise as nonlinear least squares problems. They are solved by special Gauss-Newton methods, which, in turn, require the numerical solution of linear least squares problems at each iteration. For pedagogical reasons, the order of presentation of these topics is reversed. First, linear least squares problems are discussed, including the important issue of automatic detection of rank deficiencies in matrix factorization. This topic reflects the important fact that not all data sets are equally well suited to fit all parameters of a given model. Second, the class of "adequate" nonlinear least squares problems is defined, both theoretically and computationally, for which the local Gauss-Newton method converges. Globalization via a damping strategy is presented. The case that non-convergence occurs is treated in detail to find out which part originates from an insufficient model and which one is from "bad" initial guesses for the Gauss-Newton iteration. In the final section, all pieces of the three chapters are glued together to apply to the ODE models, which are the general topic of the book. First, the notorious predator-prey problem is revisited, which turns out to be quite standard. Next, in order to connect the advocated computational ideas with modelling intuition, a simple illustrative example is worked out in algorithmic detail. Lastly, a more complex parameter identification problem related to a model of the human menstrual cycle is discussed in detail.

The **Appendix** contains mathematical background material postponed to maintain the basic flow of the presentation throughout the text. Topics selected therein are (i) the complex exponential function, (ii) condition numbers of linear algebra problems, (iii) details about QR-factorization with column pivoting, and (iv) convergence results for Gauss-Newton methods. These topics require a bit more mathematical knowledge; interested readers may find hints on further reading.

In the final section, **Software** mentioned throughout the book is listed together with web addresses, from which these codes can be downloaded.

Chapter 1
ODE Models for Systems Biological Networks

This chapter presents basics of mathematical modelling in systems biology. In Sect. 1.1, a brief introduction to the topic is given, mainly in terms of examples such as problems from population dynamics or from drug administration. In Sect. 1.2, the assembly of large ODE networks from simple chemical and physiological mechanisms, given in terms of chemical reaction modules, is described. Reasons are given, why the so-called Michaelis-Menten kinetics is no longer needed in the numerical simulation of such systems. For reaction diagram parts, where only the properties "stimulating" or "inhibiting" are known, the formulation in terms of Hill functions is presented. Finally, in Sect. 1.3, necessary mathematical background material is collected as far as it seems important for the class of applications in question. Main topics are the uniqueness and sensitivity of solutions as well as asymptotic stability. Mathematical contents are typically explained by examples rather than by theorems, while emphasis is laid on consequences for practical calculations.

1.1 Introduction

To start with, Sect. 1.1.1 gives a short overview about ODE initial value problem types that occur in systems biology. Next, two simple model problems are worked out in some detail, one from population dynamics (Sect. 1.1.2), one on multiple dose administration of drugs (Sect. 1.1.3).

1.1.1 Problem Types in Systems Biology

Let us first give a brief list of problems that typically come up in systems biology. In the subsequent Sects. 1.1.2 and 1.1.3 we will present a few elementary examples.

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0_1

Non-autonomous Initial Value Problems

This book predominantly focuses on initial value problems for systems of d ordinary differential equations (ODEs)

$$y' = f(t, y), \quad y(t_0) = y_0 \in \mathbb{R}^d \tag{1.1}$$

for given initial values y_0. The notation indicates that the time variable t appears *explicitly* in the right-hand side f; in this case we speak of a *non-autonomous* problem. However, apart from special problems of drug administration where the time point of administration enters crucially into the modelling, this case is the non-standard case in systems biology.

Autonomous Initial Value Problems

Throughout the book we will mainly deal with the case, when the time variable t does *not explicitly* enter into the right-hand side f. Then we have

$$y' = f(y), \quad y(0) = y_0 \in \mathbb{R}^d \,. \tag{1.2}$$

A specialty of this type of problem is that for any given solution trajectory $y(t)$ satisfying (1.2) there exists a continuum of further solution trajectories $z(t) = y(t - \tau)$ with time shift τ satisfying the same ODE

$$z'(t) = y'(t - \tau) = f(y(t - \tau)) = f(z(t))$$

and the same initial condition

$$z(t_0 + \tau) = y(t_0 + \tau - \tau) = y(t_0) = y_0 \,.$$

Due to this so-called *translation invariance* the initial point t_0 can be chosen arbitrarily, so that we are free to set $t_0 = 0$ in (1.2).

Parameter Dependent Problems

In the majority of problems in systems biology, a (possibly large) number of unknown parameters $p = (p_1, \ldots, p_q)$ enters in the form

$$y' = f(y, p), \quad y(0) = y_0 \in \mathbb{R}^d, \; p \in \mathbb{R}^q \,. \tag{1.3}$$

Of course, one would like to identify such parameters by matching the above type of model with given experimental data. The corresponding mathematical problem is

far more subtle than often recognized in systems biology literature. Because of its central importance in modelling, it will be carefully elaborated in Chap. 3 below.

Linear ODEs

In the non-autonomous case, a linear system may be written as

$$y' = A(t)y + b(t) \quad y(t_0) = y_0, \quad A \in \mathbb{R}^{d\times d} ,$$

where $A(t)$ denotes a time dependent (d, d)-matrix and $b(t) \in \mathbb{R}^d$ a corresponding vector function. In the autonomous case, we will most often encounter the *homogeneous* situation $b = 0$ so that

$$y' = A\, y, \quad y(0) = y_0$$

in terms of some time-independent (d, d)-matrix. This kind of system plays a role in stability analysis of general ODE systems, see Sect. 1.3.3.

Singularly Perturbed Systems

In the mathematical literature for systems biology, now and then so-called singularly perturbed problems of the kind

$$y' = f(y, z), \quad 0 = g(y, z), \tag{1.4}$$

arise. Such systems have been designed in the early days of computational science to be able to solve them by standard explicit integrators. The approach is more or less dispensable, since today efficient so-called *stiff* integrators are available that solve such problems, see Sect. 1.3.4 for a more detailed discussion.

Delay or Retarded Differential Equations

Quite often processes do not just depend on the current state but also on the "history" of the system. Such systems also arise as a phenomenological description, when not enough information about a chain of intermediate processes is at hand.

In the simplest case such a differential system contains a *retardation* or *delay time* $\tau > 0$ so that

$$y' = f\left(y(t), y(t-\tau)\right), \quad y(t) = \Theta(t) \text{ for } [-\tau, 0] \tag{1.5}$$

with a given *initial function* Θ. In contrast to the standard ODE case we typically have

$$y'(0^-) = \Theta'(0) \neq f(y(0), y(-\tau)) = y'(0^+) \ ,$$

i.e., the derivative of the solution is *discontinuous* at the initial point $t = 0$. The discontinuity propagates along the trajectory, but is gradually smoothed. This feature has to be taken into account in the numerical simulation! Typical for systems biology is the fact, that the delay may depend on the solution, too, which means that one should write $\tau(y)$ above instead of just τ. Throughout the book we will not go into too much detail of this problem type, but give a hint on available codes in Sect. 2.5.1.

Periodic ODE Problems

In systems biological modelling, *internal clocks* or *circadian rhythms* play an important role. The modelling of such processes leads to ODE problems of the kind (mostly autonomous)

$$y' = f(y), \quad y(T) = y(0) \ , \tag{1.6}$$

with *unknown period* T. In contrast to the initial value problems mentioned so far, this problem is of *boundary value type*, which is more complex and beyond the scope of this book. Interested readers may want to look up theoretical and algorithmic details in the textbook [15, Section 7.3].

1.1.2 Example: Population Dynamics

This kind of mathematical model describes the dynamics of populations. Let $p(t)$ denote the number of individuals at time t and Δt some finite time step. Then the change of population p within time interval $[t, t + \Delta t]$ will be

$$p(t + \Delta t) - p(t) = g\Delta t, \quad g \in \mathbb{R} \ ,$$

which means that the longer the time interval, the greater the change will be. Note that $g > 0$ represents growth, $g < 0$ decay. The typical derivation step now is to write down the above relation as a difference equation and pass to the limit as follows:

$$\underbrace{\frac{p(t + \Delta t) - p(t)}{\Delta t} = g}_{\text{difference equation}} \quad \stackrel{\Delta t \to 0}{\longrightarrow} \quad \underbrace{\frac{dp(t)}{dt} = g}_{\text{differential equation}}$$

This differential equation (ODE) may also be written as $p' = g$. Note that here we have tacitly applied some *continuum hypothesis* assuming that $p(t) \in \mathbb{R}^+$, even though $p(t) \in \mathbb{N}$, since the number of individuals can be counted. In addition, ODE models are based on the assumption of *well-mixing*, i.e. individuals are homogeneously distributed in space such that spatial gradients can be neglected in the model. We now turn to some special cases for the rate coefficient $g = g(t, p)$.

Exponential Growth

We start with the assumption of a constant fertility rate λ_0 (interpretation: the more individuals, the higher the birth rate):

$$p' = \lambda p, \quad p(0) = p_0 \geq 0 \,. \tag{1.7}$$

This is a *linear* ODE, which can be solved to yield

$$p(t) = p_0 \exp(\lambda t) \,.$$

The case of growth occurs with $\lambda_0 > 0$. In this case, there would be only members of this species after some time in the corresponding local neighborhood – obviously ignoring the limited nutrition basis or any other environmental constraints.

Saturation Model

The insufficiency of a purely linear model has already been pointed out 1838 by P.-F. Verhulst [61], who suggested to modify the ODE in the form

$$p' = \kappa p(p_{\max} - p), \quad p(0) = p_0 \geq 0, \quad \kappa > 0 \,. \tag{1.8}$$

Obviously, this is a *nonlinear* ODE, often also called *logistic equation*. The associated dynamics has two *fixed or stationary points* with $p' = 0$, namely $p \equiv 0$, which can only occur for $p_0 = 0$, and $p \equiv p_{\max}$, which is approached by any trajectory with $0 < p_0 \leq p_{\max}$. An illustration is given in Fig. 1.1.

As one of the rare examples, the initial value problem (1.8) can be solved analytically by *separation of variables* (let $t_0 = 0$, since we have an autonomous ODE):

$$\begin{aligned} \kappa \int_0^t dt &= \int_{p_0}^{p} \frac{dp}{p(p_{\max} - p)} \\ \kappa p_{\max} t &= \int_{p_0}^{p} \left(\frac{1}{p} + \frac{1}{p_{\max} - p} \right) dp \quad \text{(partial fraction decomposition)} \end{aligned}$$

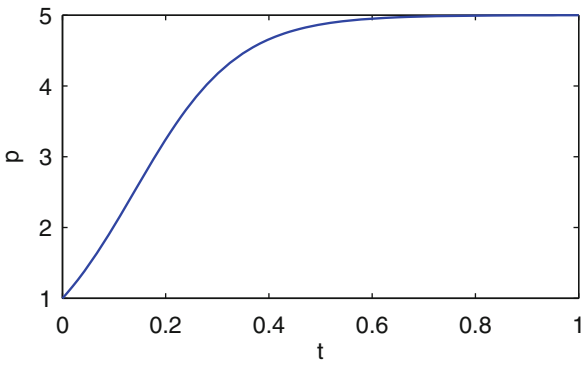


Fig. 1.1 Logistic growth ($p_0 = 1$, $\kappa = 2$, $p_{\max} = 5$)

$$\kappa p_{\max} t = (\ln p - \ln(p_{\max} - p))\Big|_{p_0}^{p} = \ln \frac{p(p_{\max} - p)}{p_0(p_{\max} - p_0)}$$

After some short calculation we obtain the analytic solution

$$p(t) = p_0 \, \frac{p_{\max}}{p_0 + (p_{\max} - p_0)\exp(-\kappa p_{\max} t)} \tag{1.9}$$

This function is often also called the *logistic law of growth*. Note that for $t = 0$ one actually obtains the initial value $p(0) = p_0$. Moreover, one easily verifies that $p(t) \le p_{\max}$ and $\lim_{t\to\infty} p(t) = p_{\max}$, if $0 < p_0 < p_{\max}$.

Predator-Prey Model

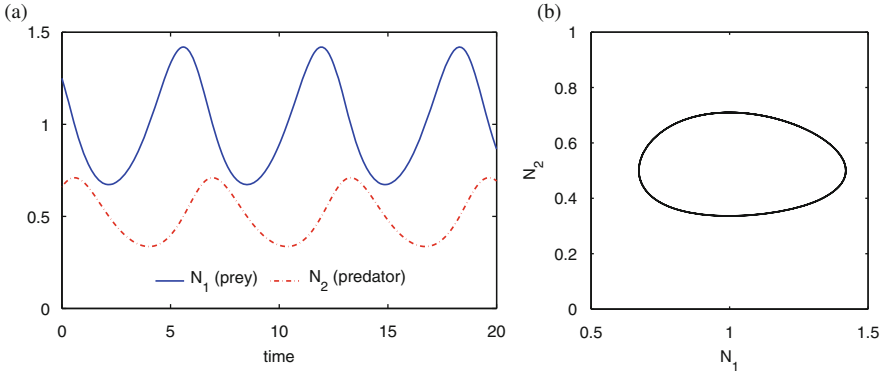
Consider the dynamics of a closed ecological system, in which *two* species interact, predators (number N_2) and prey (number N_1); as an example, you may take fox for the predator and hare for the prey. The behavior can be described by the model

$$N_1' = N_1(\alpha - \beta N_2), \quad N_2' = -N_2(\gamma - \delta N_1) \tag{1.10}$$

with prescribed positive parameters $\alpha, \beta, \gamma, \delta$. This pair of first-order nonlinear differential equations is known as *Lotka-Volterra model*, named after A. J. Lotka and V. Volterra, who independently developed these equations already in 1925/1926, see, e.g., the textbook [45] by J. D. Murray.

The nonlinear terms $N_1 N_2$ enter, since the prey population would grow unboundedly, if the predator population were zero, while the predator population would die out, if the prey population were zero. The meaning of the parameters is:

- α: prey reproduction rate (with unbounded nutrition resources),
- β: rate at which prey is eaten by predators (per unit prey), which is equivalent to mortality rate of prey per unit predator,



Solution over time. Phase plane plot.

Fig. 1.2 Predator-prey model (1.10) for parameters $\alpha = 1,\ \beta = 2,\ \gamma = 1,\ \delta = 1$, and initial value $(N_1(0), N_2(0)) = (1.25, 0.66)$

- γ: mortality rate of predators in the absence of prey,
- δ: reproduction rate of predators per unit prey.

A short calculation yields

$$\frac{dN_1}{dN_2} = \frac{N_1(\alpha - \beta N_2)}{-N_2(\gamma - \delta N_1)} \quad \Rightarrow \int \frac{-\gamma + \delta N_1}{N_1} dN_1 = \int \frac{\alpha - \beta N_2}{N_2} dN_2 \ .$$

Upon integrating both sides, we arrive at

$$-\gamma \ln(N_1) + \delta N_1 = \alpha \ln(N_2) - \beta N_2 + \text{constant} \ .$$

As a consequence, the quantity

$$H(N_1, N_2) = -\gamma \ln(N_1) + \delta N_1 - \alpha \ln(N_2) + \beta N_2 = H(N_1(0), N_2(0))$$

is an invariant along any trajectory. In Fig. 1.2, left, two oscillatory solution curves $N_1(t), N_2(t)$ are depicted. Figure 1.2, right, shows $H(N_1, N_2) = \text{const}$ in an (N_1, N_2)-plane, also called *phase plane*, where a closed orbit arises for each initial value.

In Sect. 3.5.1 below we treat the identification of parameters from given data of the Canadian lynx (predator) and snowshoe hare (prey).

1.1.3 Example: Multiple Dose Administration of Drugs

We follow the presentation in the illustrative book of D. S. Jones et al. [40] to show how drug concentrations in body fluids can be described by differential equations.

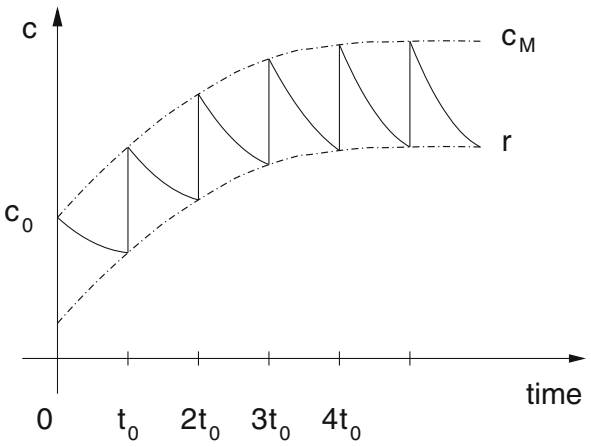


Fig. 1.3 Typical plasma concentration of a drug in multiple dose treatment

Assume that the drug concentration $c(t)$ within the blood plasma can be described by the following simple law:

$$c' = -c/\tau.$$

In this linear ODE, the constant τ, often called *relaxation time*, characterizes the decay rate of the concentration

$$c(t) = c_0 \exp(-t/\tau) \ .$$

Suppose now that some prescribed constant dose c_0 is administered regularly at times $t_n = nt_0, \ n = 0, 1, \ldots$. Then the concentration will grow in a sawtooth pattern, which is illustrated in Fig. 1.3.

Let us now try to model this situation quantitatively. For that purpose, we introduce the notation $c_n = c(t_n)$. Due to the above decay law, we get

$$c(t_n^-) = c_{n-1} \exp(-t_0/\tau)$$

and with the regular administration eventually

$$c_n = c(t_n^-) + c_0 = c_{n-1} \exp(-t_0/\tau) + c_0 \ .$$

For convenience of writing we introduce the quantity $q = \exp(-t_0/\tau) < 1$ and thus arrive at the recursion

$$c_n = c_{n-1} q + c_0 \ ,$$

from which we obtain (check yourself)

$$c_n = c_0\,(1 + q + q^2 + \ldots + q^n) = c_0\,\frac{1 - q^{n+1}}{1 - q}$$

or, in the original notation,

$$c_n = c_0 \frac{1 - \exp(-(n+1)t_0/\tau)}{1 - \exp(-t_0/\tau)}.$$

In addition, we obtain the so-called *concentration residue*

$$r_n = c(t_{n-1}^-) = c_0 \exp(-t_0/\tau)\frac{1 - \exp(-nt_0/\tau)}{1 - \exp(-t_0/\tau)}.$$

Taking the limit $n \to \infty$, we observe that the concentration never exceeds

$$c_{\max} = \frac{c_0}{1 - \exp(-t_0/\tau)},$$

and that the residue approaches

$$r = c_{\max} \exp(-t_0/\tau) = \frac{c_0}{\exp(t_0/\tau) - 1}.$$

Usually, the therapeutic goal is to reach $c_{\max}$ in only a few dose steps (t_0/τ large), whereas r should be kept above a certain level (t_0/τ small). Obviously, these two goals are in contradiction to each other. One strategy is to avoid the sawtooth build-up by giving an initial large dose of $c_0 + r$ or $c_{\max}$ and thereafter again doses of c_0. The optimal treatment strategy, however, usually depends on several factors like production costs and patterns of human behavior.

1.2 ODE Systems from Chemical or Physiological Networks

In systems biology, typical ODE systems originate from *chemical kinetics*. Apart from these, so-called *compartment* models arise, which we will explain in Example 1 below and, in a more realistic setting, subsequently in Sect. 2.5.3. In Sect. 1.2.1, we start with isolated simple chemical mechanisms and their translation into ODE models. Such models will comprise the building blocks of large networks whose construction we will discuss in Sect. 1.2.3 below. In between, in Sect. 1.2.2, we discuss some traditional model type for enzyme kinetics called Michaelis-Menten kinetics, which is still around in the literature, but is no longer needed nowadays.

1.2.1 Elementary Chemical Mechanisms

Part of the presentation here closely follows Section 1.3 in the textbook [16].

Monomolecular Reaction

In chemical language, this reaction is written in terms of two chemical species A, B as

$$A \longrightarrow B$$

In a *particle* model we may denote $n_{A,B}$ as the number of particles of A, B. In Boltzmann's kinetic gas theory, which needs to be carefully discussed when applied within the human body (under the assumptions of constant pressure, volume V, and temperature T!), one obtains for the changes $\Delta n_{A,B}$ of particle numbers $n_{A,B}$ within some time interval Δt

$$\Delta n_A \sim -n_A \Delta t, \quad \Delta n_B = -\Delta n_A$$

where the second equation is the conservation of particles. In a *continuum* model, the associated *concentrations* are defined as

$$c_A = \frac{n_A}{V}, \quad c_B = \frac{n_B}{V} \ .$$

For ease of writing, one usually identifies the names for the concentrations with the names of the corresponding chemical species, i.e. $c_A \to A, c_B \to B$ etc. Upon defining k as a reaction rate coefficient, we thus arrive at the ODEs

$$A' = -kA, \quad B' = kA \ . \tag{1.11}$$

If we set initial conditions

$$A(0) = A_0, \quad B(0) = 0 \ ,$$

then we can solve these simple equations analytically to obtain

$$A(t) = A_0 \exp(-kt), \quad B(t) = A_0(1 - \exp(-kt))$$

In passing we note that *mass conservation* still holds in the two equivalent forms

$$A'(t) + B'(t) = 0 \quad \Leftrightarrow \quad A(t) + B(t) = A(0) + B(0) = A_0 \ . \tag{1.12}$$

Bimolecular Reaction

In chemical language, this reaction reads

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C + D\,.$$

Using the same kinetic reaction principles as before, one is led to the ODE model (again identifying species and concentration names)

$$A' = B' = -k_1 AB + k_2 CD, \quad C' = D' = +k_1 AB - k_2 CD\,. \tag{1.13}$$

In passing we again note that conservation of mass holds:

$$A' + B' + C' + D' = 0. \tag{1.14}$$

Important special cases of this mechanism to arise in systems biology are

- catalysis: $B = C$
- autocatalysis: $B = C = D$,
  e.g., DNA replication: nucleotide + DNA $\rightleftharpoons$ 2 DNA

Stationary state. The equilibrium phase, also called the stationary state, is characterized by

$$A' = B' = C' = D' = 0\,.$$

From this we arrive at the classical *law of mass action kinetics* (often called the *Arrhenius law*) :

$$\frac{AB}{CD} = \frac{k_2}{k_1} = \exp\left(-\frac{\Delta E}{RT}\right) =: k_{21} \tag{1.15}$$

where we have already inserted the Boltzmann formula with ΔE the activation energy, which is the energy difference between reactants and products, R the universal gas constant, and T the temperature (as above). If only the equilibrium phase of this reaction is to be modeled, then the above equilibrium coefficient $k_{21} = k_2/k_1$ is the only degree of freedom that is well-defined. In this case, a *model reduction* is possible. A simple illustrative example for this phenomenon will be worked out in Sect. 3.5.2.

General Reaction Scheme

For the sake of completeness, we mention that a reaction of the general type

$$m_A A + m_B B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} m_C C + m_D D$$

would give rise to the *equilibrium* relation (the general law of mass action kinetics)

$$\frac{A^{m_A} B^{m_B}}{C^{m_C} D^{m_D}} = \frac{k_2}{k_1} = \exp\left(-\frac{\Delta E}{RT}\right) . \tag{1.16}$$

A general reaction of the type

$$m_1^{\text{in}} A_1 + m_2^{\text{in}} A_2 + \ldots + m_d^{\text{in}} A_d \xrightarrow{k} m_1^{\text{out}} A_1 + m_2^{\text{out}} A_2 + \ldots + m_d^{\text{out}} A_d$$

results in the reaction rate equation

$$y' = k\nu \prod_{i=1}^{d} \frac{y_i^{m_i^{\text{in}}}(t)}{m_i^{\text{in}}!},$$

where

$$y = (A_1, \ldots, A_d)^T, \quad \nu = (m_1^{\text{out}} - m_1^{\text{in}}, \ldots, m_d^{\text{out}} - m_d^{\text{in}})^T.$$

Remark 1 Often, the factorials in the above denominators are absorbed into the constant k, giving rise to a reaction rate equation in the form

$$y' = k\nu \prod_{i=1}^{d} y_i^{m_i^{\text{in}}}(t).$$

Both forms can be found in the literature; note that the value of the reaction rate coefficient will vary accordingly.

Remark 2 Whenever the copy numbers of species involved in a chemical reaction get small, random fluctuations come into play. In this case, the ODE models based on mass action kinetics must be replaced by the *chemical master equation* (CME). The CME is the fundamental equation of stochastic chemical kinetics. This differential-difference equation (continuous in time and discrete in the state space) describes the temporal evolution of the probability density function for the states of a chemical system. The state of the system represents the copy numbers of interacting species, which are changing according to a list of possible reactions. The solution of

the CME in higher dimensions is mathematical challenging and the topic of ongoing research. A detailed discussion would go beyond the scope of this book.

Inhibitory or Stimulatory Impact

In quite a number of chemical reactions in biology detailed knowledge about the individual reaction mechanisms is not available, but only some information of the kind "inhibitory or stimulatory impact". This qualitative insight is usually captured quantitatively in terms of so-called *Hill functions*. Let S denote some input substrate concentration and P the corresponding output product concentration. Then, in terms of threshold values T, T^-, T^+ and Hill coefficients n, the following modelling schemes are in common use (see Fig. 1.4):

- *Inhibitory processes.* These are described by *negative feedback* Hill functions (with the notation $X = S/T$)

$$h^-(S,T,n) = \frac{1}{1+X^n}, \quad P' = p^- h^-(S,T,n)\ , \tag{1.17}$$

  where p^- denotes some reaction rate coefficient.
- *Stimulatory processes.* These are described by *positive feedback* Hill functions (with the notation $X = S/T$)

$$h^+(S,T,n) = \frac{X^n}{1+X^n}, \quad P' = p^+ h^+(S,T,n)\ , \tag{1.18}$$

  where p^+ is again some reaction rate coefficient.

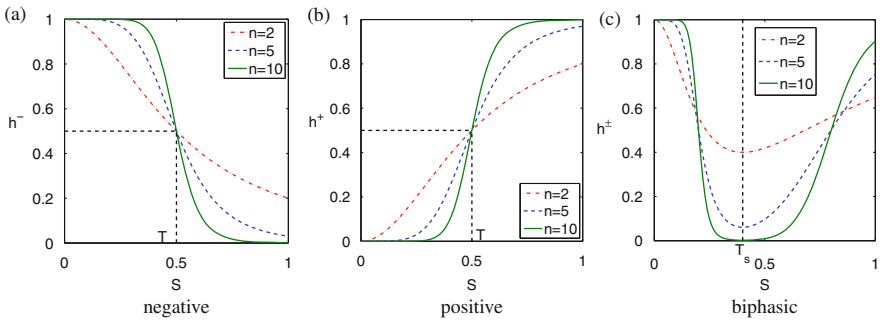


Fig. 1.4 Hill functions for variable parameter n. *Left*: negative feedback ($T = 0.5$). *Center*: positive feedback ($T = 0.5$). *Right*: switch behavior between mutually independent process directions ($T^- = 0.2, T^+ = 0.8, T_s = 0.4$)

- *Switch processes.* Whenever two process directions are mutually independent, then they can be modeled by *biphasic* Hill functions

$$h^{\pm}(S, T^{-}, T^{+}, n) = h^{-}(S, T^{-}, n) + h^{+}(S, T^{+}, n) \ , \tag{1.19}$$

which gives rise to the ODE parts

$$P' = p^{\pm} h^{\pm}(S, T^{-}, T^{+}, n) \ ,$$

with $p^{\pm}$ as reaction rate coefficient. The switch takes place at $T_s = \sqrt{T^{-}T^{+}}$, compare Fig. 1.4. In passing we note that

$$h^{-}(S, T, n) + h^{+}(S, T, n) = 1 \ .$$

Whenever the two process directions are mutually dependent, they should be coupled multiplicatively, i.e.

$$h^{\pm}(S, T^{-}, T^{+}, n) = h^{-}(S, T^{-}, n) \times h^{+}(S, T^{+}, n) \ . \tag{1.20}$$

1.2.2 Enzyme Kinetics

A special case of reaction mechanism is the case of enzyme kinetics, which we here give in some detail, since this mechanism can be treated numerically in different ways. This mechanism involves four chemical species: substrate S, product P, enzyme E, and complex C. In chemical language, this kind of reaction scheme is written as

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \xrightarrow{k_2} E + P \ .$$

The corresponding mathematical formulation in terms of an ODE system is:

$$\begin{aligned} S' &= -k_1 ES + k_{-1} C \\ E' &= -k_1 ES + k_{-1} C + k_2 C \\ C' &= k_1 ES - k_{-1} C - k_2 C \\ P' &= k_2 C \end{aligned}$$

Observe that all parameters above enter linearly, compare the remarks in Sect. 3.4 in the context of parameter sensitivity analysis. As initial conditions we typically have

$$E(0) = E_0, \ S(0) = S_0, \ C(0) = 0, \ P(0) = 0 \ .$$

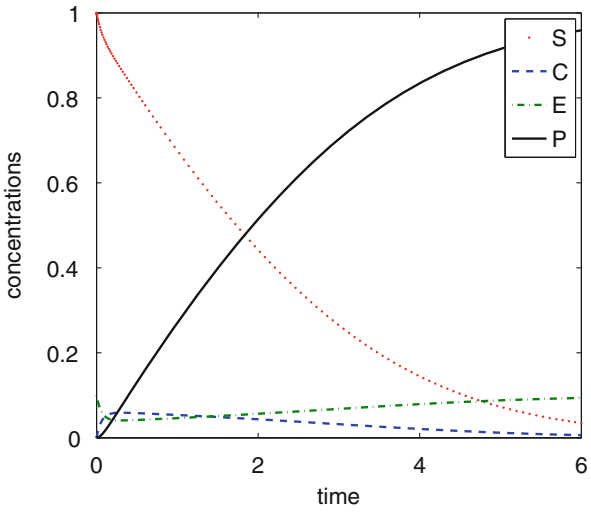


Fig. 1.5 Numerical simulation of an enzyme reaction (substrate S, product P, enzyme E, complex C)

As for mass conservation, we now have two chemical reactions:

$$E' + C' = 0 \;\Rightarrow E + C = \text{const} = E_0$$

$$S' + C' + P' = 0 \;\Rightarrow S + C + P = \text{const} = S_0\,.$$

Upon eliminating $E = E_0 - C$ and $P = S_0 - S - C$ from the above four ODEs, we obtain a reduced model with only two ODEs

$$S' = -k_1(E_0 - C)S + k_{-1}C \tag{1.21}$$

$$C' = k_1(E_0 - C)S - (k_{-1} + k_2)C \tag{1.22}$$

to be completed by the two above initial conditions $S(0) = S_0$, $C(0) = 0$. In Fig. 1.5, we show the results of numerical simulations.

Michaelis-Menten kinetics. We continue with an analysis of the above enzyme reaction mechanism by introducing the so-called *quasi-steady state approximation*, in short: QSSA. In this framework, we set

$$C' = 0\,.$$

Insertion into the ODE (1.22) then yields

$$C(t) = \frac{E_0 S(t)}{K_m + S(t)}, \quad K_m := (k_{-1} + k_2)/k_1$$

where K_m denotes the so-called *Michaelis constant*. Inserting this expression into (1.21) leads to

$$S'(t) = -k_2 E_0 \frac{S(t)}{K_m + S(t)} \,. \tag{1.23}$$

The ODE (1.23) is called *Michaelis-Menten kinetics*.

Generally speaking, the advent of modern stiff integrators (see Sects. 2.3 and 2.4) has made the QSSA including the Michaelis-Menten kinetics superfluous. Nevertheless such models have survived even in recent literature, which is why they are also accepted as possible mechanisms in the modelling language SBML [11].

1.2.3 Assembly of Large ODE Networks

The previous two sections have shown that there exists a one-to-one correspondence between elementary chemical reactions and ODE schemes. In actual systems biological modelling such small blocks will have to be assembled to large chemical reaction networks. For this purpose, it is convenient to construct a so-called *chemical compiler* that *automatically* generates the ODE system. (We deliberately skip here the possible addition of further physiological mechanisms that give rise to ODEs of different kind; they will need an extra treatment.)

Chemical Compiler

Such a programming tool generates the right-hand sides f of an ODE system (1.2) from elementary pieces. This is a comparatively easy task, since the mechanisms of Sect. 1.2.1 lead to known functions such as polynomials or Hill functions. Simultaneously, anticipating Sect. 1.3.2 below, the Jacobians f_y, f_p (with respect to variables y and parameters p) will be needed. In the polynomial terms the parameters usually enter linearly, which is why we explicitly advise users to avoid any Michelis-Menten kinetics (see (1.23)) wherever possible, since they would give rise to parameters entering nonlinearly. Of course, any additional right-hand sides originating from other source terms should be treated aside. In particular, approximation of the Jacobians f_y, f_p by numerical differentiation might be applicable, which requires special software, see [5].

Such a compiler permits a user to concentrate on modelling questions without getting too much involved with the arising ODE system. At the same time it helps to reduce programming errors. That is why already in the 1980s FORTRAN codes like CHEMKIN due to [41] or LARKIN due to [4, 22] have been developed, mainly oriented towards physical chemistry. Nowadays, CHEMKIN is developed by the

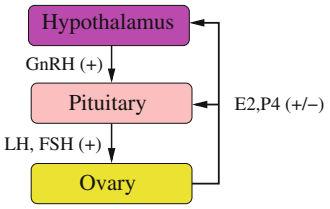


Fig. 1.6 Compartment model for the human menstrual cycle. The model maps the hypothalamic-pituitary-gonadal (hpg) axis, along which the various hormones act

company ReactionDesign,[1] whereas an open-source version, named Cantera,[2] is developed by the group of Dave Goodwin at the California Institute of Technology. More recent developments oriented towards systems biology are the SBML package [46] to be combined with numerical codes like Copasi due to [39] or BioPARKIN due to [25].

Compartment Modelling

This modelling technique is quite popular in computational biology. It consists in splitting the system under consideration into separate *compartments*, which are coupled by ODEs that describe the quantitative connections between these parts of the model. Within each of the compartments, concentrations are assumed to be uniformly distributed. Rather than discussing this technique abstractly, we illustrate it below by a recent elaborate example. In addition, we present the results of assembling chemical reaction mechanisms. Thus it may stand for a class of typical examples in systems biology. Moreover, in Sect. 2.5.3, we work out a larger compartment model concerning cancer cells.

Example 1 In [53], a model of the human menstrual cycle has been worked out in detail. The selected compartments are: the hypothalamus, the pituitary, and the ovaries, connected by the blood stream, as illustrated in Fig. 1.6.

In Fig. 1.7, part of the corresponding chemical model is presented in the usual form of a reaction diagram. The species have been colored according to their occurrence in different compartments. The full model comprises 33 chemical species (and, of course, the same number of ODEs) as well as 76 chemical reactions and physiological processes. For mere illustration purposes, we just give a selection out of the rather large compiled ODE system.

Luteinizing Hormone (LH):

$$Syn_{\mathrm{LH}}(t) = (b_{Syn}^{\mathrm{LH}} + k_{\mathrm{E2}}^{\mathrm{LH}} \cdot h^{+}(\mathrm{E2}(t), T_{\mathrm{E2}}^{\mathrm{LH}}; n_{\mathrm{E2}}^{\mathrm{LH}})) \cdot h^{-}(\mathrm{P4}(t), T_{\mathrm{P4}}^{\mathrm{LH}}; n_{\mathrm{P4}}^{\mathrm{LH}}) \tag{1.24}$$

[1] http://www.reactiondesign.com/

[2] http://cantera.github.io/docs/sphinx/html/index.html

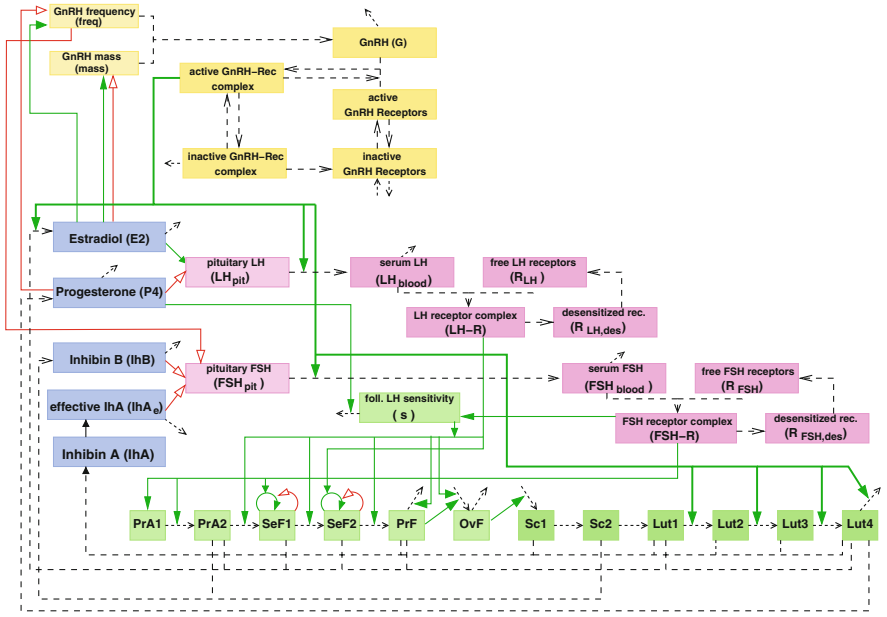


Fig. 1.7 Flowchart of a model for the human menstrual cycle. *Dashed lines*: transitions, elimination, or chemical reactions. *Solid lines with filled arrows*: stimulatory effects (Hill functions h^+). *Solid lines with unfilled arrows*: inhibitory mechanisms (Hill functions h^-). Compare this flowchart with the ODEs presented below

$$Rel_{\mathrm{LH}}(t) = \left(b_{Rel}^{\mathrm{LH}} + k_{\mathrm{G\text{-}R}}^{\mathrm{LH}} \cdot h^+(\mathrm{G\text{-}R}(t) + \mathrm{Ago\text{-}R}(t), T_{\mathrm{G\text{-}R}}^{\mathrm{LH}}; n_{\mathrm{G\text{-}R}}^{\mathrm{LH}})\right) \cdot \mathrm{LH}_{\mathrm{pit}}(t) \tag{1.25}$$

$$\frac{d}{dt}\mathrm{LH}_{\mathrm{pit}}(t) = Syn_{\mathrm{LH}}(t) - Rel_{\mathrm{LH}}(t) \tag{1.26}$$

$$\frac{d}{dt}\mathrm{LH}_{blood}(t) = \frac{1}{V_{blood}} \cdot Rel_{\mathrm{LH}}(t) - (k_{on}^{\mathrm{LH}} \cdot \mathrm{R}_{\mathrm{LH}}(t) + k_{cl}^{\mathrm{LH}}) \cdot \mathrm{LH}_{blood}(t) \tag{1.27}$$

LH receptor binding:

$$\frac{d}{dt}\mathrm{R}_{\mathrm{LH}}(t) = k_{recy}^{\mathrm{LH}} \cdot \mathrm{R}_{\mathrm{LH},des}(t) - k_{on}^{\mathrm{LH}} \cdot \mathrm{LH}_{blood}(t) \cdot \mathrm{R}_{\mathrm{LH}}(t) \tag{1.28}$$

$$\frac{d}{dt}\mathrm{LH\text{-}R}(t) = k_{on}^{\mathrm{LH}} \cdot \mathrm{LH}_{blood}(t) \cdot \mathrm{R}_{\mathrm{LH}}(t) - k_{des}^{\mathrm{LH}} \cdot \mathrm{LH\text{-}R}(t) \tag{1.29}$$

$$\frac{d}{dt}\mathrm{R}_{\mathrm{LH},des}(t) = k_{des}^{\mathrm{LH}} \cdot \mathrm{LH\text{-}R}(t) - k_{recy}^{\mathrm{LH}} \cdot \mathrm{R}_{\mathrm{LH},des}(t) \tag{1.30}$$

We deliberately dropped the equations for the gonadotropin releasing hormone (GnRH, already left out in Fig. 1.7), for the follicle stimulating hormone (FSH),

the physiological mechanisms for the development of various stages of follicles and corpus luteum as well as the reaction mechanisms for estradiol (E2), progesterone (P4) and the two inhibins (IhA,IhB). Readers interested in all details may want to look up the original paper [53].

1.3 Mathematical Background for Initial Value Problems

From the vast mathematical background material concerning ODE *initial value problems* we here want to select only such items that need to be understood when modelling and simulating networks in systems biology. In the following we will treat questions of uniqueness, sensitivities, condition numbers, and asymptotic stability.

1.3.1 Uniqueness of Solutions

Given an ODE model, it should be clear whether this model has a *unique* solution. If this were not the case, then any "good" numerical integrator will run into difficulties. That is why we discuss the topic here. Let $y^*(t), t \in [t_0, t_+[$ denote a unique solution existing over the half-open interval $t_0 \leq t < t_+$.

Uniqueness Criteria

As worked out in mathematical textbooks (see again, e.g., [16, Section 2.2]), there are three cases that may occur, from which we select two that may come up in systems biological modelling:

(a) The solution y^* exists "forever", i.e. $t_+ = \infty$.
(b) The solution "blows up" after finite time, i.e. $t_+ < \infty$.

Case (a) essentially requires that the right-hand side f satisfies a *global Lipschitz condition*

$$\|f(x) - f(y)\| \leq L\|x - y\| \quad \text{for all} \quad x, y,$$

wherein the term 'global' means that it holds for all arguments x, y. Typically, this so-called *Lipschitz constant* L is identified via the derivative of the right-hand side, to be denoted by

$$f_y(y) = D_y f(y) = \frac{\partial f}{\partial y} .$$

The expression f_y is often called the *Jacobian (matrix)* of the right-hand side. With this definition the Lipschitz constant can be calculated as

$$L = \sup_y |f_y(y)| \,, \tag{1.31}$$

where the maximum (supremum sup) is taken over all possible arguments y. This seemingly only theoretical quantity will play an important role later in connection with the definition of "stiffness" of ODEs, see Sect. 2.1.4. For illustration purposes, we give two scalar examples of the above cases.

Example 2 (Case (a)) Consider an example similar to the monomolecular reaction (1.11),

$$y' = -ky, \quad y(0) = y_0, \quad k > 0 \,.$$

The right-hand side is *linear* so that $|f_y(y)| = k$. There exists a *global* Lipschitz constant $L = k$ and thus a unique solution over all times, in the special case

$$y^*(t) = y_0 \exp(-kt) \,.$$

As $k > 0$, the solution is bounded for all $t \geq 0$.

Example 3 (Case (b)) Consider the nonlinear example,

$$y' = y^2, \quad y(0) = 1 \,,$$

similar to the bimolecular reaction (1.13). Here we obtain $|f_y(y)| = 2|y|$, which is only bounded, if we restrict the values of y. Thus we have only *local* Lipschitz continuity of f. In fact, by solving this equation analytically (using separation of variables), we see that there exists a unique solution

$$y^*(t) = \frac{1}{1-t}, \quad -\infty < t < 1,$$

only up to some finite time $t_+ = 1$. In Fig. 1.8 we give the graph of the solution.

Remark 3 In systems biology, such a Lipschitz condition will typically only hold *locally*, i.e. for restricted arguments y, which would formally allow for case (b) as well. However, due to *mass conservation* (see the examples (1.12) or (1.14)) case (b) can be excluded, since any bounded sum of positive terms assures that each term is bounded. In actual modelling, some scientists ignore mass conservation – with the danger that then solutions may "blow up". In addition, note that in numerical simulation things turn already to be bad when the solution only "nearly" blows up. Such events occur, e.g., in the realistic example in Sect. 3.5.3, where there is no mass conservation in the model.

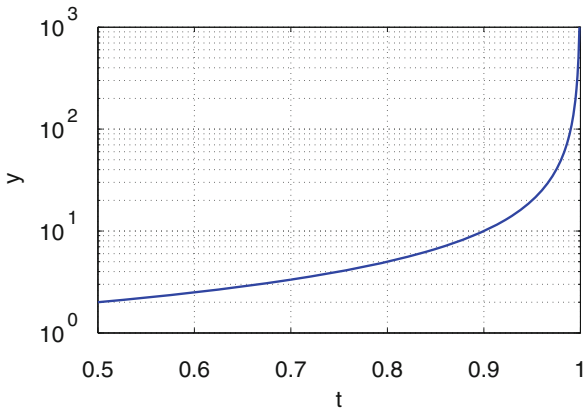


Fig. 1.8 Solution graph for Example 3

Phase Flow and Evolution

Suppose a linear system of equations were given, say $Ax = b$. If it has a unique solution, say x^*, then this can be written as $x^* = A^{-1}b$. The definition of the matrix inverse A^{-1} is just a clean notation to indicate the uniqueness of the solution; by no means should the linear equation be solved by first computing the matrix inverse and then multiply it to the right-hand side b.

In a similar way, a notation to indicate that an ODE initial value problem has a unique solution, say y^*, has emerged. For an *autonomous* initial value problem

$$y' = f(y), \quad y(0) = y_0 \,, \tag{1.32}$$

we write

$$y^*(t) = \Phi^t y_0$$

in terms of some *phase flow* (often just called *flow*) Φ^t satisfying a *semigroup* property

$$\Phi^{t-t_1}\Phi^{t_1} y_0 = \Phi^t y_0, \quad t_1 \in [0, t] \,.$$

For a *non-autonomous* IVP

$$y' = f(t, y), \quad y(t_0) = y_0$$

the unique solution y^* is defined via the *evolution* Φ^{t,t_0} as

$$y^*(t) = \Phi^{t,t_0} y_0 \,.$$

The evolution satisfies the *semigroup* property

$$\Phi^{t,t_1}\Phi^{t_1,t_0}y_0 = \Phi^{t,t_0}y_0, \quad t_1 \in [t_0, t] \,. \tag{1.33}$$

The notations $\Phi^t y_0$ and $\Phi^{t,t_0}y_0$ should not be misunderstood: these mappings are *nonlinear* functions of the initial values y_0. As in the case of the matrix inverse for linear equations, these notations should not be regarded as a recipe to solve the given ODE problem.

1.3.2 Sensitivity of Solutions

In a first step, we want to study the effect of a perturbation of the initial value y_0 in the form

$$y_0 \mapsto y_0 + \delta y_0 \,.$$

Propagation Matrices

In the *autonomous* case, the question is how this deviation propagates along the solution $y(t) = \Phi^t y_0$. In order to study this propagation, let us start with Taylor's expansion with respect to the initial perturbation δy_0, i.e.

$$\Phi^t(y_0 + \delta y_0) - \Phi^t y_0 = D_y\Phi^t y|_{y=y_0}\delta y_0 + D_y^2\Phi^t y|_{y=y_0}[\delta y_0, \delta y_0] + \dots \,.$$

Upon dropping terms of second and higher order in δy_0, we arrive at some *linearized* perturbation theory

$$\Phi^t(y_0 + \delta y_0) - \Phi^t y_0 \doteq D_y\Phi^t y|_{y=y_0}\delta y_0 =: \delta y(t) \,,$$

wherein the notation $\doteq$ denotes the linearization. The thus defined perturbation $\delta y(t)$ is given by the *linear* mapping

$$\delta y(t) = W(t)\delta y_0 \tag{1.34}$$

in terms of the (d, d)-matrix

$$W(t) := D_y\Phi^t y|_{y=y_0} = \frac{\partial y(t)}{\partial y_0} \quad \Rightarrow \quad W(0) = I_d$$

called the *propagation matrix* or *Wronskian matrix*. This matrix can be interpreted as the *sensitivity* of the nonlinear mapping Φ^t with respect to the initial value y_0.

Just like in the nonlinear case (1.33), we get some *semigroup* property

$$W(t-t_1)W(t_1) = W(t), \quad t_1 \in [0,t] \; . \tag{1.35}$$

For *non-autonomous* IVPs, we merely modify the definition of the Wronskian matrix by expanding the notation to

$$W(t,t_0) := D_y \Phi^t y|_{y=y_0} = \frac{\partial y(t)}{\partial y(t_0)} \quad \Rightarrow \quad W(t,t) = I_d \text{ for all } t$$

and thus obtain the analogous linear relation

$$\delta y(t) = W(t,t_0)\delta y_0 \; . \tag{1.36}$$

The corresponding *semigroup* property reads

$$W(t,t_1)W(t_1,t_0) = W(t,t_0), \quad t_1 \in [t_0,t] \; . \tag{1.37}$$

Variational Equation

Starting from (1.34) we may derive an ODE for the perturbation according to

$$\delta y' = W'(t)\delta y_0 \; .$$

Upon recalling the definition of the propagation matrix, we find that

$$W'(t) = f_y(\Phi^t y_0)W(t), \quad W(0) = I_d \; . \tag{1.38}$$

Insertion of this ODE above then yields

$$\delta y' = f_y(\Phi^t y_0)W(t)\delta y_0 = f_y(\Phi^t y_0)\delta y$$

The thus arising linear ODE

$$\delta y' = f_y(\Phi^t y_0)\delta y, \quad \delta y(0) = \delta y_0 \tag{1.39}$$

is called the *variational equation*. Note that this equation is *non-autonomous* due to the time dependent argument in the derivative matrix f_y. Its formal solution is (1.34), which shows that the Wronskian matrix is just the flow (or evolution, respectively) of the variational equation. Note that (1.38) is just the variational equation for the Wronskian matrix itself.

For the *non-autonomous* case $y' = f(t,y)$, we would obtain the modified variational equation

$$\delta y' = f_y(\Phi^t y_0, t)\delta y + f_t(\Phi^t y_0, t), \quad \delta y(0) = \delta y_0 \; . \tag{1.40}$$

Analogously to the autonomous case, Eq. (1.36) supplies the solution of this non-autonomous variational equation.

Condition Numbers

With the above preparations, we are now ready to define the condition of initial value problems. Recall from introductory textbooks on Numerical Analysis (such as [17]) that the condition of a problem is independent of any algorithm applied to solve it. There are two basic possibilities depending on the focus of interest. For notation, we introduce $|\cdot|$ as the modulus of the elements of a vector or matrix to be well distinguished from $\|\cdot\|$, the norm of a vector or matrix.

(a) Assume one is interested only in the solution $y(t)$ at a specific time t. Then the *pointwise condition number* $\kappa_0(t)$ may naturally be defined as the smallest number for which

$$|\delta y(t)| \leq \kappa_0(t) \cdot |\delta y_0| \ . \tag{1.41}$$

On the basis of (1.34), we thus arrive at the definition

$$\kappa_0(t) = \|W(t)\| \ , \quad \kappa_0(0) = 1 \ . \tag{1.42}$$

(b) If one is interested in the entire course of the solution $y(t)$ on the whole time interval $[0, t]$, then the *interval condition number* $\kappa[0, t]$ may be defined as the smallest number for which

$$\max_{s\in[0,t]} |\delta y(s)| \leq \kappa[0, t] \cdot |\delta y_0|$$

which then implies

$$\kappa[0, t] = \max_{s\in[0,t]} \kappa_0(s) \ . \tag{1.43}$$

The above semigroup property (1.37) directly leads to the following relations:

(i) $\kappa[0, 0] = 1$,
(ii) $\kappa[0, t_1] \geq 1$,
(iii) $\kappa[0, t_1] \leq \kappa[0, t_2], \quad 0 \leq t_1 \leq t_2$
(iv) $\kappa[0, t_2] \leq \kappa[0, t_1] \cdot \kappa[t_1, t_2], \quad 0 \leq t_1 \leq t_2$

The role of the local condition number can be seen in the following example.

Example 4 For the famous *Kepler problem*, which describes the motion of two bodies (say Earth-Moon) in a gravitational field, one may show that

$$\kappa_0^{\text{Kepler}}(t) \sim t \ , \tag{1.44}$$

which is a mild increase. The situation is very different in *molecular dynamics*, see, e.g., [16, Section 1.2], where one obtains

$$\kappa_0^{\text{molecular dynamics}}(t) \sim \exp(t/t_{\text{crit}}), \qquad t_{\text{crit}} \approx 100\,\text{fs} = 10^{-13}\,\text{s}\,. \tag{1.45}$$

This means that after some very small critical time t_{crit} the initial value problems turn to get ill-posed. As a consequence, a different type of computational approach is necessary, called *conformation dynamics*, more recently also *Markov state modelling*, see, e.g., the survey article by P. Deuflhard and C. Schütte [21] and references therein.

The just introduced two different condition numbers will be needed below in Sect. 2.1.2, where we discuss error concepts in the numerical simulation, and in the following Sect. 1.3.3.

Parameter Sensitivities

In the majority of problems in systems biology, parameter dependent systems arise in the form

$$y' = f(y,p), \quad y(0) = y_0 \in \mathbb{R}^d,\ p \in \mathbb{R}^q\,.$$

Here we are naturally interested in the effect of perturbations

$$p \mapsto p + \delta p\,.$$

with respect to $p = (p_1, \ldots, p_q)$. For this purpose we define the *parameter sensitivities*

$$y_p = \frac{\partial y}{\partial p} \in \mathbb{R}^{d\times q}\,. \tag{1.46}$$

Upon application of the chain rule of differentiation, this quantity can be seen to satisfy a modified variational equation for each parameter component

$$y_p' = f_y(y(t),p)y_p + f_p, \quad y_p(0) = 0\,. \tag{1.47}$$

Remark 4 The actual numerical solution of any of the variational equations (1.39), (1.40), or (1.47) requires to treat an extended ODE system including the original ODE (1.32) to compute the argument within the Jacobians f_y or f_p, respectively. For certain algorithmic details see Sect. 2.5.1 below.

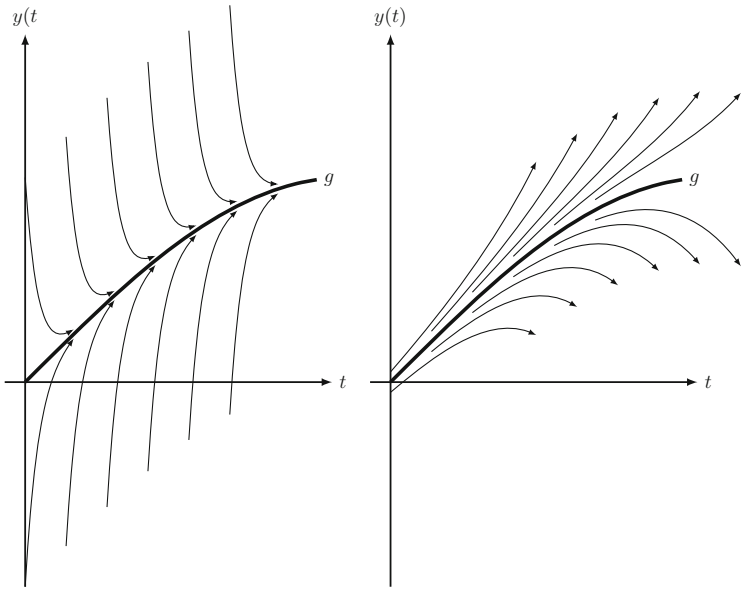


Fig. 1.9 Example 5 for an equilibrium solution $g = \sin(t), t \in [0, 1.5]$. *Left*: asymptotically stable problem (here: $\lambda = -16,\ \epsilon_0 = 1$). *Right*: inherently unstable problem (here: $\lambda = 3,\ \epsilon_0 = 0.05$)

1.3.3 Asymptotic Stability

In order to sharpen our mathematical intuition, we analyze the two types of condition numbers as introduced in the previous section for a notorious scalar ODE problem.

Example 5 Despite its simplicity this problem yields deep insight into the structure of ODEs. Let $\lambda \in \mathbb{R}$ denote some parameter in the initial value problem

$$y'(t) = g'(t) + \lambda(y - g(t)), \quad y(0) = g(0) + \delta y_0 \,. \tag{1.48}$$

The general solution may be written in the form

$$\delta y(t) := y(t) - g(t) = \exp(\lambda t)\delta y_0 \,.$$

Obviously, there exists an *equilibrium solution* $y(t) = g(t)$ for all t where g is defined. In Fig. 1.9, we give two examples for the above model problem. Upon varying the initial values y_0, we may clearly distinguish two qualitatively different situations, *asymptotic stability* versus *inherent instability*.

Condition numbers. Let us exemplify the two condition numbers defined above. From definition (1.41) and (1.42) we immediately obtain the *pointwise* condition number

$$\kappa_0(t) = |\exp(\lambda t)| \,,$$

from which (1.43) yields the *interval* condition number, say $\kappa[0,T]$ over an interval $[0,T]$. There are three qualitatively different situations for the two characteristic numbers:

(a) $\lambda < 0$: Here we get

$$\kappa_0(t) = \exp(-|\lambda|t) \overset{t\to\infty}{\longrightarrow} 0 \,, \quad \kappa[0,T] = \kappa_0(0) = 1,$$

i.e. any initial perturbation will decay over sufficiently large time intervals, see Fig. 1.9, left; in this case, the equilibrium solution $y = g$ is said to be *asymptotically stable*;

(b) $\lambda = 0$: here we obtain

$$\kappa[0,T] = \kappa_0(T) = 1, \quad \text{for all } T \geq 0 \,,$$

i.e. any initial perturbation is preserved;

(c) $\lambda > 0$: here we get

$$\kappa[0,T] = \kappa_0(T) = \exp(\lambda T) \overset{T\to\infty}{\longrightarrow} \infty \,,$$

i.e. any perturbation grows exponentially with time, the equilibrium solution $y = g$ is *inherently unstable*, see Fig. 1.9, right.

The same three cases also appear for complex valued $\lambda \in \mathbb{C}$, if we replace λ by $\Re\lambda$ in (a), (b), (c) above.

Next, we want to study a characterization of the stability properties for two more general cases.

Matrix Exponential

Suppose we have to solve the linear homogeneous autonomous initial value problem

$$y' = Ay, \quad y(0) = y_0 \in \mathbb{R}^d \,. \tag{1.49}$$

Its *formal* solution is often written in terms of the *matrix exponential*

$$y(t) = \exp(tA)y_0 \,.$$

The careful reader will observe that the matrix exponential is just the Wronskian matrix for the special case (1.49), i.e.

$$W(t) = \exp(tA)$$

In [57], a list of algorithms for the evaluation of $\exp(tA)y_0$ is collected. We want to emphasize, however, that the matrix exponential, just like any phase flow in general, should preferably be understood as a *formal* representation of the solution of (1.49), not as a basis for actual computation.

The following property of the matrix exponential is most important. Let M be an arbitrary nonsingular matrix. Then one can show that

$$\exp(tMAM^{-1}) = M\exp(tA)M^{-1} \tag{1.50}$$

For principal reasons, we briefly outline the proof. Upon multiplying (1.49) by M, we get

$$\underbrace{My'}_{=:\bar{y}'} = \underbrace{MAM^{-1}}_{=:\bar{A}}\underbrace{My}_{=\bar{y}} \quad \Leftrightarrow \quad \bar{y}' = \bar{A}\bar{y},\ \bar{y}(0) = My_0\ .$$

This yields the formal solution

$$\bar{y}(t) = \exp(t\bar{A})\bar{y}(0)$$

and, after insertion of the definitions,

$$M\exp(tA)y_0 = \exp(tMAM^{-1})My_0\ ,$$

which holds for every y_0 so that (1.50) is proven.

Warning. Note that generally

$$\exp(t(A+B)) \neq \exp(tA)\exp(tB)\ ,$$

unless the so-called commutator $[A,B]_- = AB - BA$ vanishes.

Stability of Linear Homogeneous Autonomous ODEs

For simplicity, let us assume now that A is *diagonalizable*. The results also hold in the non-diagonalizable case, which, however, is skipped here, since it is rather technical. In this case there exists a matrix M such that

$$\bar{A} = M\,\mathrm{diag}(\lambda_1,\ldots,\lambda_d)M^{-1}\ .$$

Then, with $\bar{y} = My$, the ODE $y' = Ay$ decomposes into d one-dimensional ODEs

$$\bar{y}_i' = \lambda_i \bar{y}_i, \quad i = 1, \ldots, d\,, \tag{1.51}$$

from which we obtain

$$\bar{y}_i(t) = \exp(t\lambda_i)\bar{y}_i(0) \quad \Rightarrow \quad |\bar{y}_i(t)| = \exp(t\Re(\lambda_i))|\bar{y}_i(0)|\,.$$

This gives rise to the following classification:

(a) $\Re(\lambda_i) < 0 \;\Rightarrow\; |\bar{y}_i(t)| \to 0$ for $t \to \infty$,
i.e. the solution component $\bar{y}_i$ dies out asymptotically,
(b) $\Re(\lambda_i) \le 0 \;\Rightarrow\; |\bar{y}_i(t)| \le |\bar{y}_i(0)|$,
i.e. the solution component $\bar{y}_i$ remains bounded for all $t \ge 0$,
(c) $\Re(\lambda_i) > 0 \;\Rightarrow\; |\bar{y}_i(t)| > |\bar{y}_i(0)|$,
i.e. the solution component $\bar{y}_i$ blows up for $t \to \infty$.

Of course, for systems, different components of $\bar{y}$ may fall into different classes and may be mixed via the transformation matrix M. Hence, we arrive at the following *stability criteria*:

(a) The solution y is *stable*, if $\Re(\lambda_i) \le 0$ *for all* i.
(b) The solution y is *asymptotically stable*, if $\Re(\lambda_i) < 0$ *for all* i.
(c) The solution is *unstable*, if the condition $\Re(\lambda_i) > 0$ holds for at least one index i; in this case, the instability will occur in at least one direction.

Stability of Nonlinear ODEs Around Fixed Points

We now consider general nonlinear autonomous ODEs. As shown above, linearized perturbations δy are governed by the variational equation (1.39). This equation is linear, but *non-autonomous*, i.e. of the type

$$\delta y' = A(t)\delta y, \quad \text{where} \quad A(t) = f_y(y(t))$$

with $y(t)$ the given solution to be studied. As a consequence, the above stability classification does *not* apply. Counter-examples, where the eigenvalues of some matrix $A(t)$ satisfy the above stability criterion for all t, but the perturbations $\delta y(t)$ nevertheless blow up, can be found in the literature (see the notorious example of H. O. Kreiss, e.g., [16, Remark 3.29]).

For this reason, a simpler approach studies the behavior of the solution around some fixed point $y^* \in \mathbb{R}^d$ defined by $f(y^*) = 0$. Upon defining initial values

$$y(0) = y^* + \delta y_0$$

we arrive at the variational equation,

$$\delta y' = f_y(y^*)\delta y, \quad \delta y(0) = \delta y_0 \ .$$

Obviously, in this case the Jacobian matrix $A := f_y(y^*)$ is autonomous so that the above stability theory applies.

Recall, however, that we have used a *linearized* perturbation analysis. Caution against blind application of such a theory is strongly advised. For illustration, we give the following warning example.

Example 6 We compare two simple nonlinear initial value problems.

(a) The ODE is given by

$$y' = -y^3 \ .$$

Its solution for arbitrary initial value y_0 is

$$y(t) = \left(\frac{1}{y_0^2} + 2t\right)^{-1/2} \overset{t\to\infty}{\longrightarrow} 0 \ ,$$

i.e. the system returns from any given y_0 to the fixed point $y^* = 0$. Hence, it is *asymptotically stable*.

(b) This time the ODE is given by

$$y' = y^3 \ .$$

Its solution is

$$y(t) = \left(\frac{1}{y_0^2} - 2t\right)^{-1/2} \overset{t\to t_+}{\longrightarrow} \infty \ ,$$

i.e. the system blows up at $t_+ = 1/(2y_0^2)$. Hence, it is *unstable*.

Observe, however, that both systems have the same variational equation $\delta y' = 0$ around the fixed point $y^* = 0$, which implies that $\delta y(t) = \text{const}$, even though the qualitative behavior of the two ODEs is very different. Consequently, the linearized perturbation analysis may be misleading in predicting the qualitative behavior of a nonlinear initial value problem.

1.3.4 Singularly Perturbed Problems

Assume that a given ODE system has a solution $y = (u, v)$ that naturally splits into a "slow" component u and a "fast" component v. Such a system may be written as

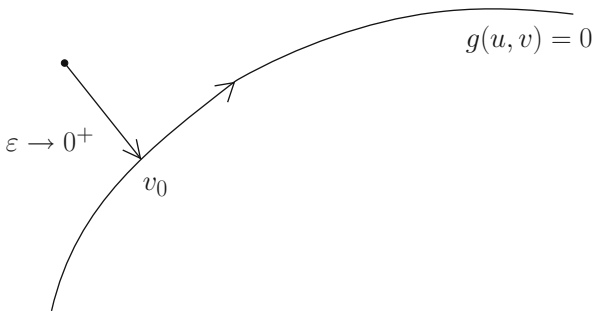


Fig. 1.10 Singular perturbation problem: Immediate approach from arbitrary starting value $v_0(0)$ to a point v_0 on the manifold $g(u, v) = 0$

a two-component system of the kind

$$u'_\varepsilon = f(u_\varepsilon, v_\varepsilon), \quad \varepsilon\, v'_\varepsilon = g(u_\varepsilon, v_\varepsilon) \tag{1.52}$$

where some "fast" time scale $\tau = t/\varepsilon$ with $0 < \varepsilon \ll 1$ has been introduced such that

$$\frac{\mathrm{d}v}{\mathrm{d}(\tau)} = \frac{\mathrm{d}v}{\mathrm{d}(t/\varepsilon)} = \varepsilon \frac{\mathrm{d}v}{\mathrm{d}t} = \varepsilon\, v'$$

Assume further that

$$\Re(\lambda(g_v)) < 0 \tag{1.53}$$

and let $\varepsilon \to 0^+$. Then, in the quasi-steady state approach (abbreviated: QSSA), we obtain the differential-algebraic equation (DAE)

$$u'_0 = f(u_0, v_0), \quad 0 = g(u_0, v_0) \tag{1.54}$$

for some two-component solution $y_0 = (u_0, v_0)$. Due to (1.53) we may interpret the limit in such a way that, for arbitrary starting value $v_0(0)$, the solution component v_0 "immediately" approaches a near-by value on the constraint manifold. For illustration, see Fig. 1.10.

With the availability of modern adaptive *stiff integrators* (see Chap. 2 below), there typically is no visible performance difference between the numerical solution of the ODE (1.52) and of the DAE (1.54). In fact, while the ODE system usually has a unique solution, the DAE may not have a unique solution, unless further assumptions hold. Instead of diving into theoretical details (to be found, e.g., in the textbook [16] and references therein) we give an illustrative example from reaction kinetics.

Example 7 This example treats a chemical network that models the thermal decomposition of n-hexane. The system comprises 47 chemical reactions for 25 chemical species, i.e. there are $d = 25$ ODEs. Among the 25 species, chemical

insight may identify 13 species as chemically stable, while 12 are so-called "free radicals". Hence, in a first QSSA treatment of the kind (1.54) (reported in [18]), one might be tempted to come up with 13 ODEs and 12 algebraic equations. In this case there exists no unique solution – as can be proven mathematically precisely; this result also came out by application of the stiff integrator LIMEX, which is equipped with a special *uniqueness monitor* (skipped here). If only 7 of the radicals are selected for the algebraic equations (after some trial and error), then a unique solution exists. However, the computing times are the same as without any QSSA preprocessing. These results are arranged in Table 1.1.

Table 1.1 Computing time comparison of two QSSA approaches for an ODE system originating from 47 chemical reactions for $d = 25$ species (due to [18]). Chemical insight would indicate a subset of 12 species regarded as "free radicals", which mathematically leads to a problem that does not have a unique solution

25 ODE's	0.48 sec.
13 ODE's, 12AE's	no unique solution exists
18 ODE's, 7 AE's	0.49 sec.

Chapter 2
Numerical Simulation of ODE Models

In the preceding chapter we had worked out how to establish possibly large ODE models for systems biological networks. In the present chapter, we deal with their numerical simulation. For this purpose, we describe various numerical integrators for initial value problems in necessary detail. In Sect. 2.1, we present basic concepts to characterize different discretization methods. We start with local versus global discretization errors, first in theory, then in algorithmic realization. Stability concepts for discretizations lead to an elementary pragmatic understanding of the term "stiffness" of ODE systems. In the remaining part of the chapter, different families of integrators such as Runge-Kutta methods, extrapolation methods, and multistep methods are characterized. From a practical point of view they are divided into explicit methods (Sect. 2.2), implicit methods (Sect. 2.3), and linearly implicit methods (Sect. 2.4), to be discussed in terms of their structural strengths and weaknesses. Finally, in Sect. 2.5, a roadmap of numerical methods is given together with two moderate problems that look rather similar, but require different numerical integrators. Moreover, we present a more elaborate example concerning the dynamics of tumor cells; therein we show, what kind of algorithmic decisions may influence the speed of computations.

2.1 Basic Concepts

Throughout this section we consider the *numerical integration* of in general nonlinear ODE initial value problems. For ease of writing, we confine our interest to autonomous problems

$$y' = f(y), \quad y(0) = y_0 \,, \tag{2.1}$$

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0_2

unless explicitly stated otherwise. In Sect. 2.1.1, we explain two concepts of discretization errors at the simplest possible example, the classical Euler method. These concepts show up in the control of local discretization errors and their relation to the actually achieved final error, to be presented in Sect. 2.1.2. As the ODE systems arising from systems biology are typically large and "stiff", we derive stability concepts in Sect. 2.1.3 that help to understand this term. For practical applications, some rather pragmatic "definition" of stiffness is given in the final Sect. 2.1.4.

2.1.1 *Local Versus Global Discretization Error: Theoretical Concepts*

In this section, we discuss selected elementary discretization schemes in terms of the *timestep* or *step size* τ. We begin with the classical scheme discussed by Leonhard Euler even before the invention of the concept of differential equations.

Classical (Explicit) Euler Method

The first idea of this scheme is to discretize the ODE (2.1) by using the geometric *tangent* at the starting point

$$y_1 = y_0 + \tau f(y_0) \tag{2.2}$$

The second idea is to repeat the first step recursively as (see Fig. 2.1)

$$y_{k+1} = y_k + \tau f(y_k), \quad k = 0, 1, 2, \ldots. \tag{2.3}$$

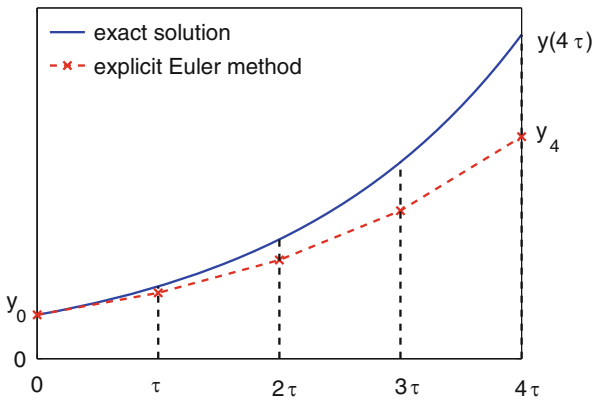


Fig. 2.1 Explicit Euler method with constant step size τ

For ease of presentation, we here stick to a *uniform* discretization with a single constant step size τ in this section. For a *non-uniform* discretization we should rather write

$$y_{k+1} = y_k + \tau_k f(y_k), \quad k = 0, 1, 2, \ldots . \tag{2.4}$$

Local Discretization Error

At the simple Euler discretization scheme we can already explain the general concepts of local and global discretization errors. After one discretization step, a deviation $y_1 - y(\tau)$ between the discrete and the continuous solution arises. In order to reveal the dependence of this deviation on τ, we use (2.1) in integrated form, i.e.

$$y(\tau) = y_0 + \int_0^\tau f(y(s))ds \, . \tag{2.5}$$

Then, together with (2.2), we are able to derive the relation

$$y_1 - y(\tau) = \tau f(y_0) - \int_0^\tau f(y(s))ds \, .$$

Taylor expansion of the above integrand yields

$$f(y(s)) = f(\underbrace{y(0)}_{y_0}) + s\underbrace{f_y(y_0)\, f(y_0)}_{=f'(y_0)} + \mathcal{O}(s^2) \, .$$

Here we have used the convenient notation $\mathcal{O}(s^p)$ for terms of order at least p. Upon inserting this expression into the integral above, the first right-hand term cancels and we arrive at

$$y_1 - y(\tau) = -\int_0^\tau [sf'(y_0) + \mathcal{O}(s^2)]ds = -\frac{\tau^2}{2} f'(y_0) + \int_0^\tau \mathcal{O}(s^2)ds \, .$$

As the integral term on the right-hand side is obviously $\mathcal{O}(\tau^3)$, we end up with a *local discretization error* estimate of the type

$$\|y_1 - y(\tau)\| \leq c \cdot \tau^2 \, , \tag{2.6}$$

where c is a constant containing information of the problem at hand (such as $f'(y_0)$).

Global Discretization Error

After N time steps according to (2.3) some *global discretization error* at *fixed* final time $T = N\tau$ will arise. Intuitively we obtain

$$\|y_N - y(T)\| \leq C \cdot N\tau^2 = C \cdot T\tau \ , \tag{2.7}$$

assuming $N \to \infty$ as $\tau \to 0$. Obviously, in the transition from local to global error we lose one order of τ.

It should be mentioned that the constants c in (2.6) and C in (2.7) are different. A rigorous proof of the result (2.7) can be found, e.g., in [16] and further references therein. As it turns out, such a proof shows that the above constant C is only bounded, if a *Lipschitz condition* of the kind

$$L\tau \leq \text{const} \tag{2.8}$$

holds, where L is the Lipschitz constant introduced in (1.31) and const means some small number, say, not much larger than 1. This is a severe *restriction on the step size* τ that holds for a large class of discretization methods and will come up at several occasions throughout this book.

Implicit Euler Method

Once the classical discretization (2.2) had been known, the question arose whether it might be useful to insert the tangent at the *new* value y_1 by virtue of

$$y_1 = y_0 + \tau f(y_1) \ . \tag{2.9}$$

Here the value y_1 is only *implicitly* defined by some in general nonlinear system of equations. That is why the above scheme is called *implicit* Euler discretization and accordingly (2.2) the *explicit* Euler discretization. Again the scheme is repeated recursively according to

$$y_{k+1} = y_k + \tau f(y_{k+1}), \quad k = 0, 1, 2, \dots \ .$$

Formally speaking, estimates for the local and global discretization error will just repeat (2.6) and (2.7). The numerical solution of the algebraic equations (dimension d) per each time step is certainly much more work than the corresponding explicit Euler recursion. For so-called "stiff" problems, however, this additional computational amount pays off, see Sect. 2.1.4 below.

Implicit Trapezoidal and Midpoint Rule

Once the door had been opened to modify the classical Euler method by introducing the implicit structure, a symmetric right-hand side has been chosen such as

$$y_{k+1} = y_k + \frac{\tau}{2}\left(f(y_k) + f(y_{k+1})\right), \quad k = 0, 1, 2, \ldots, \tag{2.10}$$

which is called the *implicit trapezoidal rule*. Another also symmetric modification is the *implicit midpoint rule*

$$y_{k+1} = y_k + \tau f\left(\frac{1}{2}(y_k + y_{k+1})\right), \quad k = 0, 1, 2, \ldots, \tag{2.11}$$

We will come back to these elementary discretizations at various occasions.

General Case

The two introduced convergence concepts, even though merely exemplified at rather simple discretizaton schemes, directly carry over to more general discretization methods to be presented below in the remaining parts of Chap. 2. For so-called *one-step methods*, the local discretization error has the typical structure

$$\|y_1 - y(\tau)\| \leq c_p \cdot \tau^{p+1} \tag{2.12}$$

in terms of some *order p* characteristic of the method, while the global error, again under some condition of the kind (2.8), satisfies

$$\|y_N - y(T)\| \leq C_p \cdot N\tau^{p+1} = C_p \cdot T\tau^p \,. \tag{2.13}$$

Note that $p = 1$ holds for the explicit as well as for the implicit Euler discretization, while $p = 2$ holds for the implicit trapezoidal rule and the implicit midpoint rule; this can be directly derived from the *symmetry* $(y_k, y_{k+1}, \tau) \leftrightarrow (y_{k+1}, y_k, -\tau)$. In all cases, a reduction of τ implies a reduction of both the local and the global discretization error bounds.

Throughout the subsequent chapters we will restrict ourselves to the first discretization step from y_0 to y_1, but tacitly include the total step from y_0 to y_N via the successive recursions from y_k to y_{k+1} for $k = 0, 1, \ldots$.

Step-Size and Order Control

Efficient modern integrators will adapt their performance to problem dependent information to choose "optimal" step sizes, say τ_k at step k, part of them also deliver

locally "optimal" orders p_k. Note that not always the rule "the higher the order the better" holds, since the above constants c_p strongly influence the achievable step sizes. In summary, order and step-size control are linked – see, e.g., the material worked out in Sects. 2.2–2.4 for each of the discussed numerical integrators.

2.1.2 *Local Versus Finally Achieved Accuracy: Algorithmic Concepts*

Throughout this section, let a prescribed fixed integration interval $[0, T]$ be subdivided according to

$$0 = t_0 < t_1 < \ldots < t_N = T \ ,$$

i.e. by $N + 1$ integration points, chosen to be non-uniformly distributed for the time being. The local and global error concepts introduced in Sect. 2.1.1 above turn out to have their correspondence in the numerical realization. For ease of notation, we define the *local discretization errors* as $\delta y_j = y(t_j) - y_j$ and require, in terms of some suitable vector norm $\| \cdot \|$, that

$$\|\delta y_j\| \leq \mathrm{TOL} \ , \tag{2.14}$$

where TOL is a user prescribed *local error tolerance*. Such errors can be controlled by all modern numerical integrators, see Sects. 2.2–2.4 below. At time point t_j, let $\delta y(t_j)$ denote the *accumulated error*. At final time T one thus has the *global discretization error*

$$\|\delta y(T)\| = \mathrm{ERR} \ , \tag{2.15}$$

wherein the value ERR usually can *not* be prescribed or controlled by numerical integrators. That is why it is of interest to study the relation between TOL and ERR in theoretical terms.

Error Propagation

In order to study the relation between local and global error, we derive a linearized theory of the error propagation, as illustrated in Fig. 2.2.
At some time step $t_j \to t_{j+1}$, we obtain the linearized relation

$$\delta y(t_{j+1}) \doteq \delta y_{j+1} + W(t_{j+1}, t_j)\delta y(t_j) \ , \tag{2.16}$$

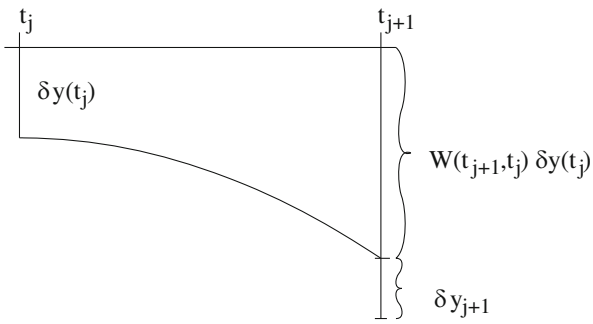


Fig. 2.2 Linearized local error propagation, see (2.16)

wherein $W(\cdot,\cdot)$ denotes the Wronskian matrix introduced in Sect. 1.3.2 above (for the non-autonomous case). By induction, starting with $\delta y_0 = 0$ and exploiting the semigroup property (1.37), here as $W(t_{j+2}, t_j) = W(t_{j+2}, t_{j+1})W(t_{j+1}, t_j)$, we arrive at

$$\delta y(T) \doteq \sum_{j=1}^{N} W(T, t_j)\, \delta y_j \tag{2.17}$$

Taking norms, we get

$$\mathrm{ERR} \doteq \sum_{j=1}^{N} \|W(T, t_j)\| \cdot \overbrace{\|\delta y_j\|}^{\leq \mathrm{TOL}} \leq \mathrm{TOL} \sum_{j=1}^{N} \kappa[t_j, T] \tag{2.18}$$

in terms of the interval condition numbers $\kappa[\cdot,\cdot]$ introduced in (1.43) above. Since N depends on TOL, there is, in general, no *linear* relation between ERR and TOL.

Role of Interval Condition Number

Obviously, the key issue above is the structure of the interval condition number. If we apply the theoretical characterization in terms of the Lipschitz constant L defined in (1.31) and return, for simplicity, to a uniform grid with $t_j = j\tau$, we arrive at the theoretical estimate

$$\kappa[t_j, T] \leq \exp(L(T - t_j)) = \exp(L(N - j)\tau) \tag{2.19}$$

and thus end up with the estimate

$$\mathrm{ERR} \doteq \mathrm{TOL} \cdot \frac{\exp(LT) - 1}{\exp(L\tau) - 1} \tag{2.20}$$

Such an estimate will only be reasonable for those ODE problems whose dynamical behavior can be characterized in terms of the Lipschitz constant; in Sect. 2.1.4 below, we will call such problems "non-stiff". Suppose, however, we have an ODE problem where

$$\kappa[t_j, T] \leq \text{const} \,, \tag{2.21}$$

which means that local errors die out asymptotically and thus dominate global errors. In mathematical analysis such problems are mostly called "dissipative", whereas in numerical analysis they are usually called "stiff". Insertion of (2.21) into (2.18) then yields

$$\text{ERR} \,\dot{\leq}\, \text{const} \cdot N \,\text{TOL} \,. \tag{2.22}$$

Total Error

Let the integration order p and the step size τ be fixed. Then, from (2.13), we roughly have the following global discretization error

$$\text{ERR} \;\doteq\; C_p T \tau^p \sim \tau^p \,.$$

Apart from the discretization errors, we will also have to deal with *rounding errors*. On a computer with relative precision eps ($\sim 10^{-16}$ typically today) we roughly obtain the contribution

$$\overline{\text{ERR}} \;\doteq\; \gamma_p N \text{eps} = \frac{\gamma_p \text{eps} T}{\tau} \sim \frac{1}{\tau}$$

with a constant γ_p depending on the discretization method. Note that for $\tau \to 0$ the discretization errors decrease, whereas the rounding errors increase, as shown schematically in Fig. 2.3. For the *total error* $\text{ERR}_{\text{total}} = \overline{\text{ERR}} + \text{ERR}$ there exists a smallest achievable accuracy ERR_{min} at τ_{opt}.

Summary

In numerical integrators only *local* accuracies can be controlled by the user prescribed error tolerance TOL. Global accuracy, say ERR, depends on the interval condition, say $\kappa[0, T]$, of the problem and can be tested by a few runs with different parameters TOL.

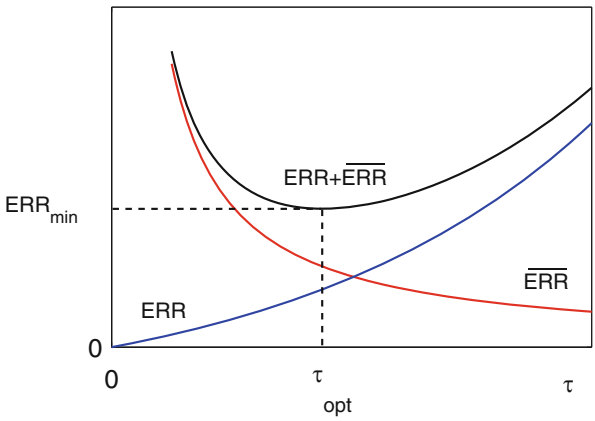


Fig. 2.3 Total error as sum of discretization error ERR and rounding error $\overline{\text{ERR}}$. The limit accuracy ERR_{min} occurs at τ_{opt}

2.1.3 Stability Concepts for Discretizations

In Sect. 1.3.3 above, we had discussed the concept of asymptotic stability of ODE initial value problems. Here, we now deal with the question of whether the properties of the continuous problem are inherited to the discretized problems. In order to understand the subsequent line of argument, the reader may want to brush up his knowledge of the complex-valued exponential function by looking up Appendix A.1.

Dahlquist Test Model

For linear autonomous systems we had found out in Sect. 1.3.3 that they boil down to d simple scalar equations of the kind (1.51). That is why G. Dahlquist [12] had suggested in 1956 to study the stability properties of discretizations by the *test problem* (today named after him)

$$y' = \lambda y, \quad y(0) = 1, \quad \lambda \in \mathbb{C}\,. \tag{2.23}$$

The analytical solution of this problem is

$$y(\tau) = \exp(\lambda\tau) = \exp(z) \quad \text{where} \quad \tau \geq 0,\ z := \lambda\tau \in \mathbb{C}\,. \tag{2.24}$$

The trick here is to formulate the stability problem in terms of the complex-valued exponential function $\exp(z)$, which we discuss in some detail in Appendix A.1. From the relations (A.2) we may deduce the basic properties:

$$\begin{cases} |\exp(z)| \geq 1 & \Leftrightarrow \quad \Re(z) \geq 0 \\ |\exp(z)| \leq 1 & \Leftrightarrow \quad \Re(z) \leq 0 \\ |\exp(z)| = 1 & \Leftrightarrow \quad \Re(z) = 0 \end{cases} \tag{2.25}$$

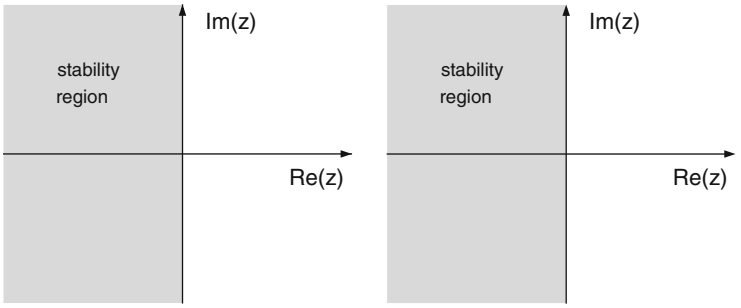


Fig. 2.4 Complex half-plane as stability region. *Left*: continuous solution. *Right*: discrete solution obtained by the implicit trapezoidal or the implicit midpoint rule

If we define the *stability region* by

$$\mathcal{S} := \{z \in \mathbb{C} : |y_1(z)| \leq 1\}\,, \tag{2.26}$$

we may identify $\mathcal{S}$ for the continuous solution with the complex half-plane (see Fig. 2.4, left)

$$\mathbb{C}_- := \{z \in \mathbb{C} : \Re(z) \leq 0\} \tag{2.27}$$

Examples of Stability Regions

For illustration, we apply the four elementary discretization schemes of Sect. 2.1.1 to the Dahlquist model (2.23). For the explicit Euler scheme we obtain

$$y_1 = y_0 + \tau f(y_0) = y_0 + \tau\lambda y_0 = 1 + \tau\lambda\,, \quad \Rightarrow \quad y_1 = 1 + z\,.$$

The corresponding stability region is shown in Fig. 2.5, left. In a similar way we get for the implicit Euler scheme

$$y_1 = y_0 + \tau f(y_1) = y_0 + \tau\lambda y_1 = 1 + \tau\lambda y_1\,, \quad \Rightarrow \quad y_1 = \frac{1}{1-z}\,.$$

The corresponding stability region is shown in Fig. 2.5, right. Finally, for the implicit trapezoidal and the implicit midpoint rule, which are equivalent for linear problems, we get

$$y_1 = y_0 + \tau(f(y_1) + f(y_0))/2 = y_0 + \tau\lambda(y_1 + y_0)/2 \quad \Rightarrow \quad y_1 = \frac{1 + z/2}{1 - z/2}\,,$$

The corresponding stability region is shown in Fig. 2.4, right.

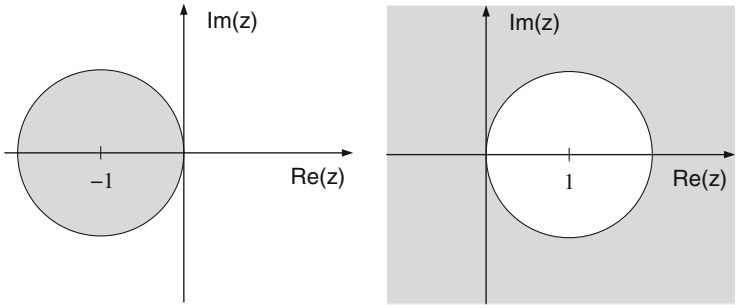


Fig. 2.5 Stability regions. *Left*: explicit Euler scheme. *Right*: implicit Euler scheme ("superstability")

From Fig. 2.4, one might think that the discretizations based on the implicit trapezoidal rule or the implicit midpoint rule are best, since they perfectly inherit the stability region from the continuous solution. However, the continuous solution has an additional desirable feature: For $z \to \infty$ one obtains

$$|\exp(z)| \quad \longrightarrow \quad \begin{cases} \infty, & \text{if } \Re(z) > 0 \\ 1, & \text{if } \Re(z) = 0 \\ 0, & \text{if } \Re(z) < 0 \end{cases}$$

If we compare this limit property for the elementary discretizations, we see that

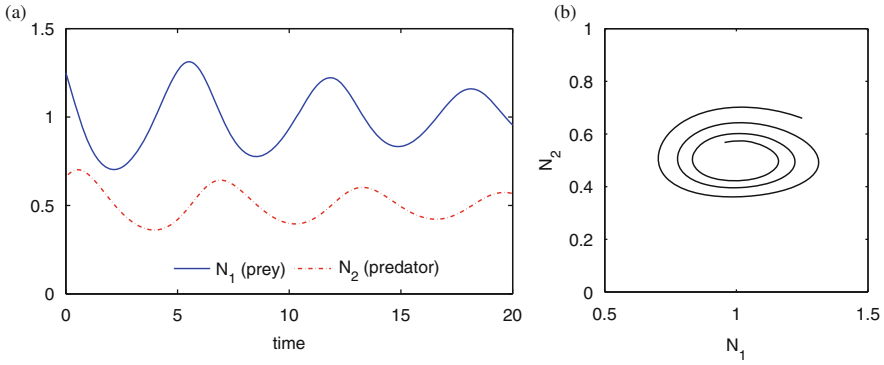
$$|y_1(z)| \quad \longrightarrow \quad \begin{cases} \infty, & \text{for the explicit Euler scheme} \\ 1, & \text{for implicit trapezoidal or midpoint rule} \\ 0, & \text{for the implicit Euler scheme} \end{cases}$$

From (A.3) we know that the limit $z \to \infty$ for the complex exponential function depends on the path taken to approach this point. Such a behavior cannot be mimicked by polynomials or rational functions, where a unique limit independent of the path of approach exists. Consequently, we cannot expect to be able to realize all of the properties of the analytical solution by a single discretization. In view of this insight, several stability concepts have been introduced to characterize desirable features of different discretizations.

A-Stability

In view of the fact that $\mathbb{C}_- = \mathcal{S}$ for the continuous solution, this concept is defined by requiring the weaker property

$$\mathbb{C}_- \subset \mathcal{S}$$



Numerical solution by implicit Euler method. Phase plane plot: shrinking of orbits.

Fig. 2.6 *Example of superstability:* Predator-prey model (1.10) as in Fig. 1.2. Qualitatively wrong solution by implicit Euler method with constant step size $\tau = 0.1$. Correct numerical solution see Fig. 1.2

for discretizations. Looking at the above Figs. 2.4 and 2.5, we directly see that the explicit Euler scheme is *not* A-stable, whereas the other three discretizations are. Note, however, that the implicit Euler scheme has the undesirable feature to supply decaying solutions even when the continuous solutions increase – associated with those parts of $\mathcal{S}$ that cover part of the positive complex half-plane. This unwanted property is called *superstability* and illustrated for the predator-prey model in Fig. 2.6.

A(α)-Stability

As it turns out, some discretizations exhibit only "almost" A-stability. In order to quantify this feature, O. Widlund [62] in 1967 introduced some angle domain (shown in Fig. 2.7)

$$\mathbb{C}(\alpha) := \{z \in \mathbb{C}\,;\, |\arg(z)| \le \alpha\}$$

and suggested the weakened concept of A(α)-stability defined via

$$\mathbb{C}(\alpha) \subset \mathcal{S}\,.$$

Observe that $\mathbb{C}(\alpha)$ permits a characterization of the complex value $z = \lambda t$, which holds, given λ, *for all* $t > 0$.

Clearly, since $\mathbb{C}(\pi/2) = \mathbb{C}_-$, it is tacitly understood that the larger α, the "more stable" the method is. Such a statement, however, should be taken with care in view of the limit feature mentioned above.

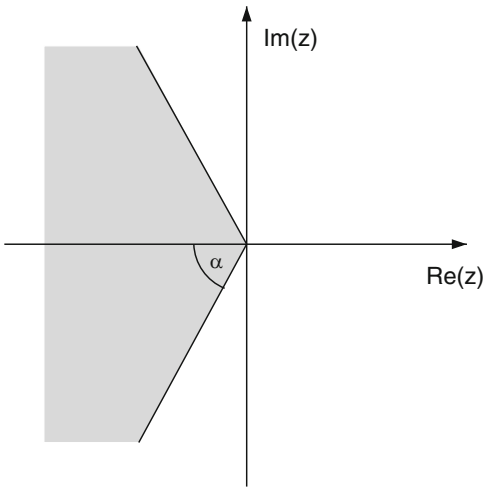


Fig. 2.7 Angle domain $\mathbb{C}(\alpha)$ for the definition of weakened stability concepts

L-Stability

This stability concept dates back to B.L. Ehle [27] in 1969. It additionally incorporates the asymptotic behavior of a discretization scheme for $z \to \infty$ by requiring

$$\mathbb{C}_- \subset \mathcal{S} \text{ and } y_1(\infty) = 0 \,. \tag{2.28}$$

In other words: A discretization method is L-stable, if it is A-stable and vanishes at $z = \infty$. Looking again back to our four sample discretizations, we observe: The explicit Euler scheme is *not* L-stable, since it satisfies neither of the two conditions. The implicit Euler scheme is L-stable, satisfying both conditions (but exhibits the unwanted superstability!). The implicit trapezoidal and the implicit midpoint rule are *not* L-stable, since $|y_1(\infty)| = 1$, which violates the second condition above. Even worse, the discrete values y_k oscillate according to

$$y_k \quad \to \quad (-1)^k \quad \text{as} \quad z \to \infty \,.$$

This asymptotic behavior is consistent with the behavior (A.3) of the continuous solution along the imaginary axis. It already shows up "close to" the imaginary axis for "large" $\Im z$ as $k \to \infty$ via the occurrence of "spurious" oscillations, i.e. via "numerical artifacts" that have may have nothing to do with the actual solution, see Fig. 2.8. For this reason, these two discretizations should only be applied, if the problem itself has eigenvalues close to the imaginary axis (and thus the solution exhibits "real" oscillations). As a rule of thumb, whenever a problem is nonlinear, then the implicit midpoint rule should be preferred over the implicit trapezoidal rule – for reasons worked out, e.g., in [16, Section 6.3.2] and references therein.

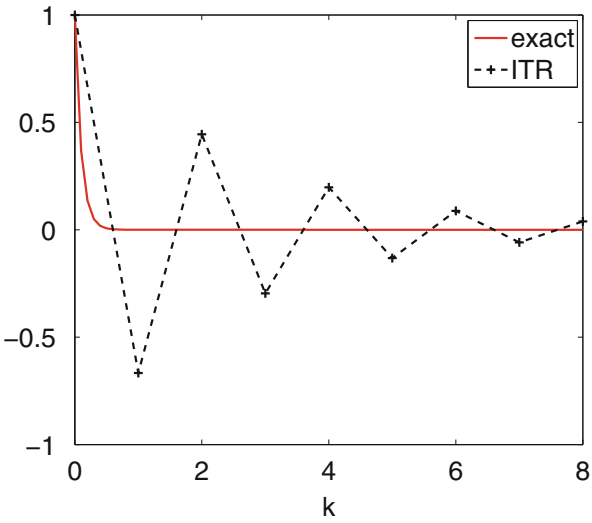


Fig. 2.8 Oscillatory behavior of the discrete values y_k obtained via the implicit trapezoidal rule with constant stepsize $\tau = 1$ applied to Dahlquists's test equation with $\lambda = -10$

L(α)-Stability

Just like A(α)-stability, a stability concept weakening L-stability exists as well, accordingly called L(α)-stability and defined via

$$\mathbb{C}(\alpha) \subset \mathcal{S} \text{ and } y_1(\infty) = 0 \, .$$

Again: the larger α, the better. Note, however, that the inclusion of the asymptotic property $y_1(\infty) = 0$ makes this concept much more useful in assessing numerical discretization schemes than mere A(α)-stability. We will return to this concept repeatedly.

2.1.4 Stiffness of ODE Problems

The stability analysis of the preceding Sect. 2.1.3 has been based on the simple Dahlquist test model,

$$y' = \lambda y, \quad y(0) = 1 \, ,$$

where $\lambda \in \mathbb{C}$ plays a rather different role depending on whether the sign of $\Re\lambda$ is positive or negative. However, if we calculate the Lipschitz constant L defined in (1.31), which is essential for all proofs of uniqueness of ODE models, we obtain

$$L = |\lambda| \, .$$

Hereby, the essential sign information is lost. Upon recalling the relation (2.8), we come up with some step-size restriction of the kind

$$\tau \dot{\leq} \frac{1}{|\lambda|}$$

Things appear differently when seen through the glass of the *pointwise condition number* defined in (1.42): For the Dahlquist model we get

$$\kappa_0(t) = \exp(\Re\lambda t) \ ,$$

which clearly maintains the necessary information of the sign of $\Re\lambda$, as can be seen in (2.25). Note that the *interval condition number* defined in (1.43) arises as

$$\kappa[0,t] = \begin{cases} \kappa_0(t), & \text{if } \Re(\lambda) > 0 \\ 1, & \text{if } \Re(\lambda) \leq 0 \end{cases}$$

In view of (2.18), we now see that

$$\text{ERR} \dot{\leq} \text{TOL} \cdot \begin{cases} \frac{\exp^{\Re\lambda T} - 1}{\exp^{\Re\lambda \tau} - 1}, & \text{if } \Re(\lambda) > 0 \\ N, & \text{if } \Re(\lambda) \leq 0 \end{cases}$$

The above factor N may reduce to essentially around 1, if the exponent is "large negative", which is the case in strongly dissipative problems, where the global error is equivalent to the local error. If we turn from the analytic solution of the Dahlquist model to a discrete solution obtained by some selected discretization scheme, then any *step-size restrictions* will show up via the condition $z = \lambda\tau \in \mathcal{S}$ in terms of the stability region $\mathcal{S}$. If, in addition, the stability region nicely models the stability region of the analytic solution, then such a constraint will be reasonable. So we fall back on the stability concepts of the Sect. 2.1.3. Summarizing, discretization methods that mimic the stability behavior of the analytic solution of the Dahlquist model will have a tolerable error propagation, whereas others do not.

Characterization of Stiffness

There are countless numbers of definitions of "stiffness" of ODE problems, see, e.g., the textbook [16] and references therein. For computational scientists that are not mathematicians, we offer the following rather pragmatic definition:
For stiff ODE problems, the additional computational amount of solving linear or nonlinear equations within each (linearly) implicit discretization step pays off, since the number of discretization steps is significantly reduced.

Note that this "definition" is rather ad hoc and will depend on the required accuracy and the selected discretization method. Practically speaking, if the nature of the problem at hand is not clear in advance, one might start with an explicit discretization; if the problem can be handled with "reasonably looking" computational step sizes, then we regard it to be non-stiff. If, however, "too strong" step-size restrictions occur, so that "too many" time steps need to be taken, see the theoretical restrictions (2.8) or (2.46), then we regard the problem to be stiff. Note that any arising small step sizes would be appropriate in the *transition phase* of an otherwise stiff problem, but awkward in the *stationary phase*. Upon observing such an undesirable behavior one will switch to a (linearly) implicit discretization method.

For a *qualitative* characterization of a given problem as *stiff*, one will examine whether the trajectory asymptotically approaches some *equilibrium point* or some *stationary smooth* solution. In this sense, most of the ODE models arising in systems biology (including all of the problems mentioned in Sect. 1.2) are stiff and thus deserve the numerical solution by some (linearly) implicit discretization method.

In order to develop some *geometric intuition*, we recall Fig. 1.9 above, where we had illustrated *stability* versus *inherent instability* around some more general stationary solution $g(t)$. For ease of understanding, let us again characterize these two cases:

- An asymptotically *unstable* or "non-stiff" problem (Fig. 1.9, right) should be attacked by some *explicit* numerical integrator, which will automatically choose small step sizes τ that are totally in order, since they reflect the dynamics of the problem.
- An asymptotically *stable* or "stiff" problem (Fig. 1.9, left) should be attacked by some *(linearly) implicit* numerical integrator, which will choose large step sizes τ that reflect the lack of dynamics of the stationary solution $g(t)$ at the computational expense of solving nonlinear systems in each discretization step; if any explicit numerical integrator were chosen in this case, extremely small step sizes would automatically be selected that would blow up computing times beyond any tolerable amount.

2.2 Explicit Numerical Integration Methods

In this section, we present three extensions of the *explicit* Euler method. The first two ones are *one-step methods*, i.e. methods that only use information from the present step to compute approximations for the next step; these include Runge-Kutta methods in Sect. 2.2.1 (dating back to 1895) and extrapolation methods in Sect. 2.2.2 (with origins back in 1910). The third one is a *multistep method*, i.e. a method that exploits the history of a trajectory to compute the next step. Among them we restrict our attention to Adams methods (dating back to 1855). Even though all of these methods seem to be rather old, there are modern *adaptive* versions that

should definitely be preferred for the numerical simulation of ODE models from systems biology.

2.2.1 Runge-Kutta Methods

Already in 1895, C. Runge suggested an improvement over the classical Euler scheme, which read

$$y_1 = y_0 + \tau f\left(y_0 + \frac{\tau}{2} f(y_0)\right)$$

Careful examination of this scheme reveals an order $p = 2$, compare definition (2.12) above, i.e. one order higher than that of the explicit Euler scheme. This increase of order has been attained by introducing an explicit Euler step with $\tau/2$ as step size inside $f(\cdot)$. In 1901, W. Kutta recognized the general nested structure to increase the order: He suggested what nowadays is called an *s-stage explicit Runge-Kutta scheme*

$$k_i = f\left(y_0 + \tau \sum_{j=1}^{i-1} a_{ij} k_j\right), \quad i = 1, \ldots, s$$

$$y_1 = y_0 + \tau \sum_{i=1}^{s} b_i k_i$$

Note the key feature that this scheme is recursive in the unknown directions k_i. Any Runge-Kutta method can be characterized by the coefficients $(b_i), (a_{ij})$, in matrix vector notation written as $(b, \mathcal{A})$, where b is an s-vector and $\mathcal{A}$ a lower triangular (s, s)-matrix with zero diagonal entries – which represents the recursive structure of the scheme.

Remark 5 When applied to *non-autonomous* systems with $f(t, y)$ as right-hand sides, the above terms are to be replaced by terms of the kind

$$f\left(t_0 + c_i\tau, y_0 + \tau \sum_{j=1}^{i-1} a_{ij} k_j\right) \quad \text{where} \quad c_i = \sum_{j=1}^{i-1} a_{ij}\,. \tag{2.29}$$

Once he had detected the general structure, W. Kutta also worked out a rather economic scheme of order $p = 4$, today named as the *classical* RK4 scheme:

$$k_1 = f(y_0)$$

$$k_2 = f(y_0 + \frac{\tau}{2} k_1)$$

$$k_3 = f(y_0 + \frac{\tau}{2}k_2)$$
$$k_4 = f(y_0 + \tau k_3)$$
$$y_1 = y_0 + \tau\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right)$$

Surprisingly, this ancient scheme is today still seen in modern computational science! However, in the meantime of more than 100 years, much more efficient higher order RK schemes have been worked out – see below.

Error Estimation and Step-Size Control

In order to simulate an ODE model reliably, the discretization error must be kept below a prescribed accuracy threshold. As worked out in Sect. 2.1.1, we have to deal with both local and global discretization errors. An estimation of the global discretization error, which means the finally achieved accuracy, would require an unreasonably large computational amount. The local discretization error, however, can be estimated via the construction of so-called *embedded* Runge-Kutta methods. In this approach, two RK methods with common coefficients $\mathcal{A}$, but different coefficients b and $\hat{b}$ are combined so that the scheme with b has order p, say, while the one with $\hat{b}$ has order $p-1$. After one step from $t = 0$ to $t = \tau$, the discretization error can be approximated according to

$$\|y_1 - y(\tau)\| \approx \|\sum_{i=1}^{s} \tau(b_i - \hat{b}_i)k_i\| =: \epsilon \tag{2.30}$$

A number of subtle considerations is necessary to back this device theoretically, see, e.g., [16, Section 5.3].

If the error of the higher order RK method with coefficients $(b, \mathcal{A})$ is estimated, the corresponding embedded RK method is usually written as RK (p-1)p, say RK (7)8. In this case, formula (2.12) combined with (2.30) above yields

$$\epsilon \approx \|y_1 - y(\tau)\| \leq c_p \tau^{p+1} \,. \tag{2.31}$$

Suppose now we want to identify an "optimal" step size τ^* such that

$$\epsilon^* \approx \|y_1 - y(\tau^*)\| \doteq c_p(\tau^*)^{p+1} \leq \mathrm{TOL} \,,$$

where TOL is a user prescribed local error tolerance. Division of the two relations then leads to the formula

$$\tau^* = \tau \sqrt[p+1]{\frac{\rho \cdot \mathrm{TOL}}{\epsilon}} \tag{2.32}$$

where we have introduced some safety parameter $\rho < 1$, usually $\rho = 0.9$, with the aim to (roughly) ensure that the thus selected step size would then actually lead to

$$\epsilon^* \leq \mathrm{TOL} \, .$$

If several RK methods of different orders are realized, then the whole device can be enriched by some *order control*, details of which we will present in Sect. 2.2.2 below in the context of some subset of RK methods.

Dense Output

If more output data are wanted than are supplied by the step-size control, then the idea of an additional embedded RK method is again exploited; generalizing (2.31) one requires that

$$\|y_1(\theta) - y(\theta\tau)\| \leq c_{\bar{p}} \tau^{\bar{p}+1} \quad \text{for} \quad \theta \in [0,1] \, . \tag{2.33}$$

The art is to find formulas that yield an order $\bar{p}$ as high as possible. Efficient RK methods usually include such a device, which also permits an extension to the numerical treatment of delay differential equations; as an example we mention the code RETARD due to E. Hairer.

Dormand-Prince Integrators

In recent decades, a whole "RK technology" for the construction of higher order RK methods has emerged, see, e.g., [16, Section 4.2.2] and references therein to the original literature. Given stage numbers s, the general principle is to determine the coefficients $(b, \mathcal{A})$ such that the discretization errors are of prescribed order p. This leads to a set of *(underdetermined) nonlinear algebraic* equations, rapidly growing with increasing orders, see Table 2.1. Note that the number of conditions only depends on the order p, not on the number s of stages, which only influences the number $(s^2 + 1)/2$ of independent coefficients to be determined. To solve these equations, additional wishes may be fulfilled concerning, e.g., embedding with more than two combined RK methods, economy of function evaluations as well as reliability and robustness of step-size control devices.

Table 2.1 Number N_p of algebraic equations for coefficients of Runge-Kutta methods depending on order p

p	1	2	3	4	5	6	7	8	9	10	20
N_p	1	2	4	8	17	37	85	200	486	1 205	20 247 374

Starting around 1980, J. R. Dormand and P. J. Prince [26] have developed a sequence of highly efficient explicit Runge-Kutta methods up to higher order, putting all theoretical and algorithmic pieces together. Their presently most efficient codes DOPRI5 and DOP853 have been economized with respect to number of function evaluations, efficiency of step-size control, dense output etc. The code DOP853 due to E. Hairer additionally realizes an automatic control of orders among the embedded orders $\{8, 5, 3\}$.

2.2.2 Extrapolation Methods

In this section, we present certain discretization methods that formally are a subset of explicit Runge-Kutta methods, but are constructed in a rather different way, along which the cumbersome solution of the many algebraic equations, see Table 2.1, can be avoided. For a general survey on extrapolation methods for (non-stiff and stiff) ODE problems we refer to [14].

Basic Procedure

The general idea of extrapolation methods is to run a simple *basic discretization* with *successively reduced internal step sizes*

$$\sigma_1 = \tau/n_1 > \sigma_2 = \tau/n_2 > \dots \quad \text{for given} \quad 1 \le n_1 < n_2 < \dots \in \mathcal{F}$$

up to the next time point τ. In this way, successively "better" approximations $y_1(\tau)$ emerge that can be regarded as functions of σ and written as $y_1(\tau;\sigma_1), y_1(\tau;\sigma_2), \dots$. These approximations serve as nodal values for polynomial interpolation over the nodes $[\sigma_1, \dots, \sigma_v]$. The interpolation polynomials $p(\sigma|\sigma_1, \dots, \sigma_v)$ are then evaluated at $\sigma = 0$, which lies outside the interpolation interval, hence the name *extra*polation, see Fig. 2.9.

Theoretical basis for such a discretization is the existence of some "polynomial-like" expansion

$$y_1(\tau;\sigma) - y(\tau) = g_1(\tau)\sigma^{\gamma_1} + g_2(\tau)\sigma^{\gamma_2} + g_3(\tau)\sigma^{\gamma_3} + \dots$$

A closer examination of such expansions for a class of discretization methods reveals that they would not converge, but can be replaced by *asymptotic expansions* of the kind

$$y_1(\tau;\sigma) - y(\tau) = \sum_{j=1}^{N} g_j(\tau)\sigma^{\gamma_j} + G_{N+1}(\tau;\sigma)\sigma^{\gamma_{N+1}} \ , \tag{2.34}$$

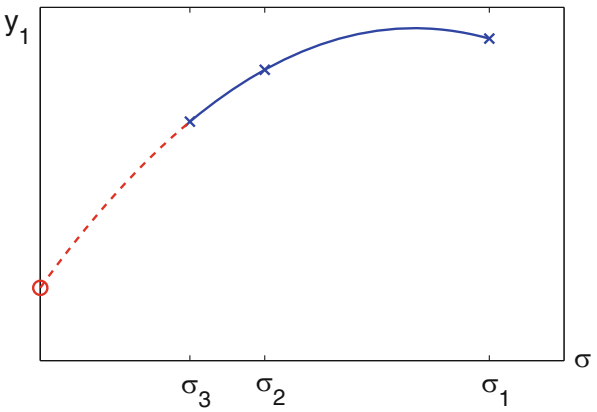


Fig. 2.9 Idea of extrapolation: Evaluation of the interpolation polynomial $p(\sigma|\sigma_1, \sigma_2, \sigma_3)$ at the limit $\sigma = 0$

Table 2.2 Aitken-Neville scheme for extrapolation methods (here $k = 3$)

$$\begin{array}{ccccc} T_{11}y & & & & \\ & \searrow & & & \\ T_{21}y & \rightarrow & T_{22} & & \\ & \searrow & & \searrow & \\ T_{31}y & \rightarrow & T_{32} & \rightarrow & T_{33} \end{array}$$

where a remainder term G takes care of the convergence properties. These expansions serve the purpose needed for extrapolation methods. Proofs of their existence including the boundedness of the remained terms are quite subtle and can be found in the usual textbooks, see, e.g., [16, Section 4.3] and references therein.

Explicit Euler Discretization

Let us explain the method at the simplest possible example, when the explicit Euler discretization is selected as the basic discretization. Starting with $\eta_0 = y_0$ the basic scheme for internal step size $\sigma_\nu = \tau/n_\nu$ reads

$$\eta_{n+1} = \eta_n + \sigma_\nu f(\eta_n), \quad n = 0, \ldots, n_\nu - 1, \quad \rightarrow \quad y_1(\tau; \sigma_\nu) = \eta_{n_\nu} \,. \tag{2.35}$$

Here the numbers n_ν are chosen from some *subdivision sequence* $\mathcal{F}$. For this basic scheme, the existence of an asymptotic expansion (2.34) with $\gamma_j = j$ can be shown. We thus may extrapolate to the limit $\sigma = 0$ using the Aitken-Neville algorithm (e.g.,[17, Section 7.1.2]). Starting with $y_1(\tau; \sigma_i) = T_{i,1}$, this scheme reads in our case

$$T_{i,k} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\frac{n_i}{n_{i-k+1}} - 1} \tag{2.36}$$

For an illustration see Table 2.2.

For the discretization error after k extrapolation steps one obtains

$$\varepsilon_{ik} := \|T_{i,k} - y(\tau)\| \doteq \|g_k'(0)\|\tau\sigma_i \cdots \sigma_{i-k+1} = \gamma_{ik}\|g_k'(0)\|\tau^{k+1} , \tag{2.37}$$

with the constant

$$\gamma_{ik} = (n_{i-k+1} \cdot n_i)^{-1} .$$

Along these lines, an extrapolation method can be designed. But there exists a "better" option to be described next.

Remark 6 Formally, this extrapolation method is a special explicit RK method with stage number

$$s = \sum_{v=1}^{k} n_v ,$$

where k is the maximum index in the extrapolation table. Its minimum as achieved for the *harmonic* subdivision sequence

$$\mathcal{F}_H = \{1, 2, 3, 4, \ldots\}$$

The present codes actually realize this sequence.

Explicit Midpoint Rule

Already in 1910, C. Richardson had suggested to apply τ^2-extrapolation by exploiting the *symmetry* of the explicit mid-point rule

$$\eta_{n+1} = \eta_{n-1} + 2\sigma_v f(\eta_n) , \quad n = 1, 2, \ldots .$$

Obviously, this recurrence cannot be realized without some value for η_1. In other words: it needs a *starting step*, a problem that Richardson had not been able to solve.

It took until 1963 that W. B. Gragg in his thesis [29] was able to prove that a simple explicit Euler starting step is enough to guarantee the existence of a *quadratic* asymptotic expansion (2.34) with $\gamma_j = 2j$. Moreover, he showed that

- there exist *two different* quadratic asymptotic expansions for n_v either *even* or *odd* and
- for an *even* subdivision sequence one additionally gains one order in τ in the approximation error.

Gragg also suggested the special final step

$$y_1(\tau;\sigma_v) := \overline{\eta}_n = \frac{1}{4}(\eta_{n-1} + 2\eta_n + \eta_{n+1}), \quad n = n_v \text{ even} . \tag{2.38}$$

This final step, which requires the additional evaluation of $f(\eta_n)$, is not really crucial, but useful for dense output, see below. Again this method is formally an explicit RK method, so that minimizing the stage number leads one to implement the *double harmonic* subdivision sequence (see [13])

$$n_v \in \mathcal{F}_{2H} = \{2, 4, 6, 8, \ldots\ldots\} .$$

Quadratic extrapolation. On this basis, one just needs to modify the Aitken-Neville algorithm (2.36) in the form

$$T_{i,k} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{n_i}{n_{i-k+1}}\right)^2 - 1} . \tag{2.39}$$

Again one gets a triangular extrapolation tableau as shown in Table 2.2. For the discretization error after k extrapolation steps one here obtains

$$\varepsilon_{ik} := \|T_{i,k} - y(\tau)\| \doteq \|g_k'(0)\|\tau\sigma_i^2 \cdots \sigma_{i-k+1}^2 = \gamma_{ik}^2 \|g_k'(0)\|\tau^{2k+1} , \tag{2.40}$$

with the same constant γ_{ik} as defined in (2.37).

Order and Step-Size Control

In order to characterize the linear versus quadratic asymptotic expansions, we write $\gamma_j = \gamma j$ with $\gamma = 1$ for the explicit Euler scheme and $\gamma = 2$ for the explicit mid-point rule. On the basis of the error formulas (2.37) and (2.40), the above formula (2.32) readily suggests "optimal" step sizes for each column index k in Table 2.2

$$\tau_k^* = \tau \sqrt[\gamma k+1]{\frac{\rho \cdot \mathrm{TOL}}{\epsilon_{k+1,k}}}, \quad \rho < 1 . \tag{2.41}$$

For strong algorithmic reasons one uses the *subdiagonal* error estimate

$$\epsilon_{k+1,k} := \|T_{k+1,k} - T_{k+1,k+1}\| \approx \|T_{k+1,k} - y(\tau)\|$$

Thus each column is associated with its own step size suggestion. As for the amount of work, this is essentially only dependent on the row i in the tableau, so that we

may denote it by A_i. In order to measure the *work per unit step* we define the computationally available quantities

$$W_k := \frac{A_{k+1}}{\tau_k}\tau = A_{k+1}\left(\frac{\epsilon_{k+1,k}}{\mathrm{TOL}}\right)^{\frac{1}{\gamma k+1}} .$$

With this set of values, we may find an "optimal" column index, say q, by requiring the computable criterion

$$W_q = \min_{k=1,\ldots k_{\max}} W_k \tag{2.42}$$

Here a "greedy algorithm" argument has been used, see [17]. Once q is determined, the step size $\tau^* = \tau_q^*$ is taken as the basic step size for the next step. (In passing we note that the same kind of idea is used to select an "optimal" order within embedded explicit RK codes like DOP853.)

In addition to this order and step-size selection, the whole approximation table T_{ik} is exploited to monitor any unwanted deviation from an expected convergence pattern, details are left to [14]. This abundance of information makes extrapolation methods extremely robust in real life applications.

Dense Output

In general, extrapolation integrators adaptively select rather large step sizes due to their efficient order and step-size control. If more output data are wanted, then an extra tool is necessary. In connection with the explicit Euler or midpoint rule, it is easy to add a Hermite interpolation tool, see [17, Section 7.1.2]. This cubic interpolation is based on the information $y_0, f(y_0)$ and $y_1, f(y_1)$. The discrete values y_1 are anyway available at the end of each τ-step; in order to get some accurate value $f(y_1)$, one will evaluate f after the extrapolation for y_1; this does not increase the overall computational amount, since this value can be stored to be used as starting value for the next step. An efficient higher order technique for dense output in explicit extrapolation methods has been work out by E. Hairer and A. Ostermann in [32]; for the explicit midpoint rule, this technique requires a change of the subdivision sequence away from $\mathcal{F}_{2H}$, which is why we here do not pursue it. For the purpose of systems biology, we are only interested to get a cheap way of evaluating data at prescribed dense output points – an aim that can be achieved to sufficient accuracy already with the cubic Hermite interpolation, given the adaptive step-size and order control.

Note, however, that along this line backward dense output information is available only *after* each τ-step; consequently, such a tool is not sufficient to extend the extrapolation method in the direction of delay differential equations, since there the interpolation values are necessary *during* the discretization.

Explicit Extrapolation Codes

An extrapolation code EULEX based on the explicit Euler scheme as basic discretization has been exemplified in [14]; this code essentially served as a model to explain the idea of an extrapolation method. Based on the explicit mid-point rule, two implementations are quite popular, DIFEX1 due to [14], which includes the above explained dense output option, and ODEX due to [34]; the two codes differ only slightly in secondary details of the above order and step-size control and are similarly efficient, both of them typically twice as fast as EULEX.

2.2.3 Adams Methods

In the two previous Sects. 2.2.1 and 2.2.2 we had presented one-step methods, i.e. methods that only use information from $t_0 = 0$ to compute an approximation y_1 at the next time step $t_1 = \tau$. Already in 1855, J. C. Adams had the idea to exploit more of the "history" of the trajectory.

General Scheme

Adams directly started from the integral form (2.5), which we repeat for convenience as

$$y(\tau) - y_0 = \int_{t=0}^{\tau} f(y(t))\, dt\ .$$

The idea is to replace the integrand f by a polynomial g_k interpolating the $k+1$ "previous" values

$$f(y_1), f(y_0), f(y_{-1}), \ldots, f(y_{-k+1})\ .$$

This interpolation polynomial is unique and so we may write

$$f(y(t)) \quad \rightarrow \quad g_k(t) = \sum_{j=0}^{k} L_j(t) f(y_{j-k+1})$$

where the L_j denote the Lagrange polynomials that depend on the interpolation nodes only, see, e.g., [17, Section 7.1]. Insertion of this expression yields a discretization scheme of the kind

$$y_1 - y_0 = \tau\,(\beta_k f(y_1) + \beta_{k-1} f(y_0) + \beta_{k-2} f(y_{-1}) + \ldots + \beta_0 f(y_{k-1})) \tag{2.43}$$

where the coefficients $\beta_k, \ldots, \beta_0$ are uniquely determined by

$$\beta_j = \frac{1}{\tau} \int_{t=0}^{\tau} L_j(t)\, dt = \int_{\theta=0}^{1} L_j(\theta\tau)\, d\theta \tag{2.44}$$

If $\beta_k = 0$, the scheme is *explicit*, in which case the polynomial g_k is of order k.

Remark 7 For readers with theoretical interest we briefly want to mention that the above Adams method is not just an arbitrary candidate out of a large class of possible multistep methods, but has an outstanding *stability* property. This stability corresponds to the numerical solution of the trivial non-stiff ODE model $y' = 0$, obviously the special case of the Dahlquist test model for $\lambda = 0$; it assures that the obtained numerical solution dominates any possibly occurring "parasitic" numerical artifacts. For more details see, e.g., [16, Section 7.3.1]. This goes with the fact that Adams methods are only useful for *non-stiff* problems.

Discretization Error Estimate

Of course, to start such a scheme requires suitable starting values. Under certain (unrealistic) assumptions on such starting values, a rough examination of these schemes yields the discretization errors

$$y_1 - y(\tau) = \mathcal{O}(\tau^p), \quad p = \begin{cases} k, & \text{if } \beta_k = 0 \\ k+1, & \text{if } \beta_k \neq 0 \end{cases} \tag{2.45}$$

To start a scheme of order $\bar{p} \geq 2$, one must implement a *start-up procedure*

$$p = 1, 2, \ldots, \bar{p} - 1\ .$$

The initial step is always an explicit Euler step. For the start-up procedure, the above error behavior (2.45) changes from step to step, i.e. the order of discretization error increases from step to step until it reaches the order level strived for.

PECE Methods

For $\beta_k \neq 0$ the scheme (2.43) is *implicit* and the order p is $k+1$, i.e. one gains one order in comparison with the explicit version. However, one now has to solve a nonlinear system for the unknown y_1. Originally, Adams had suggested to use Newton's method, while Moulton later suggested to use a fixed point iteration. Today one realizes a variant named PECE method. In this approach, one starts with the associated explicit Adams method to obtain some *predictor* y_1^P. This value is then inserted into formula (2.43), wherein the value $f(y_1^P)$ is *evaluated* (E) to supply a *corrector* value y_1^C, which, in turn, requires the *evaluation* (E) of $f(y_1^C)$. This is the

first step of a fixed point iteration, which would converge under some condition of the kind

$$L\tau \leq \text{const}\,. \tag{2.46}$$

This condition directly reminds one of the Lipschitz condition (2.8), which indicates that this extension of the explicit Adams method, too, can only be expected to be useful for *non-stiff* ODE problems.

For historical reasons, the *explicit* Adams methods are also called *Adams-Bashforth* methods (dating back to 1883), whereas the *implicit* Adams methods including the PECE methods are often named as *Adams-Moulton* methods.[1]

Order and Step-Size Control

The *adaptive* control of order and step size within any multistep method is much more difficult than in the one-step case. We do not want to go too much into details here, but refer interested readers to the textbook [16, Section 7.4]. For a potential user of Adams methods, only a few general remarks seem to be appropriate:

- The coefficients $\beta_0, \ldots, \beta_k$ introduced in (2.43) can be calculated off-line, if a *uniform* grid with constant step size τ is realized, see (2.44). If, however, some *step-size control* is realized, then this leads to a *non-uniform* grid, which, in turn, changes the Lagrange polynomials L_j in every new integration step; as a consequence, the coefficients need to be recomputed in the course of the numerical computation whenever the grid changes. For this reason, the Lagrange representation is replaced by some so-called *Nordsieck* implementation, which allows an easier handling of varying grids.
- If one wants to vary the *order*, then an alternative implementation based on Newton's divided differences is preferred.
- Due to the fact that order and step-size control require different implementations, Adams methods gain most of their efficiency on uniform grids and with constant order, i.e. when the ODE to be solved exhibits some quite regular dynamics. For the same reason, they are less reliable and robust in the efficient numerical solution of problems with rapidly changing dynamics.

Adams Codes

A rather efficient modern *adaptive* PECE code is DEABM (abbreviating Adams-Bashforth-Moulton) due to L. F. Shampine and H. A. Watts from 1979 or LSODE

[1] Moulton published his method not earlier than 1926, since it was regarded as a "military secret" during World War I.

(E stands for explicit) due to A. C. Hindmarsh [36] from 1980. All of these implementations have a natural way of realizing a "dense output" option, since they are based on interpolation.

2.3 Implicit Numerical Integration Methods

In this section, we present two types of efficient numerical *stiff integrators*, which are extensions of the *implicit* Euler method. Their implicit structure requires the numerical solution of *nonlinear* equations in each discretization step; this is realized via Newton-type methods, which require a linear equation solve in each iteration until convergence. It should be explicitly mentioned that solving these nonlinear equations by some fixed point iteration would not be appropriate, since this would assume some condition like (2.46), in contradiction to the intended treatment of stiff equations. First, in Sect. 2.3.1, we discuss *collocation methods*, the most efficient stiff integrators of implicit one-step or Runge-Kutta type. Next, in Sect. 2.3.2, we elaborate the *backward differentiation method* (in short: *BDF method*), the optimal candidate among multistep methods for stiff problems.

2.3.1 Collocation Methods

In Sect. 2.2.1, the general structure of explicit Runge-Kutta schemes has been shown. In 1963, J. C. Butcher [10] extended these schemes in an interesting way.

Implicit Runge-Kutta Schemes

According to Butcher, an *s-stage implicit Runge-Kutta scheme* is denoted by

$$k_i = f\left(y_0 + \tau \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, \ldots, s$$

$$y_1 = y_0 + \tau \sum_{i=1}^{s} b_i k_i$$

In general, such a scheme is no longer recursive in the unknown directions k_i, but requires the solution of a system of $s \cdot d$ *nonlinear* equations for $k_1, \ldots, k_s$. Any thus defined Runge-Kutta method can again be characterized by coefficients $(b, \mathcal{A})$, where b is an s-vector, but now $\mathcal{A}$ is a full (s, s)-matrix. In order to establish a general RK method of order p, one has to solve N_p algebraic equations (see Table 2.1) for the $s^2 + s$ coefficients, which gives a lot more degrees of freedom than in the merely

explicit RK case. The extension to the *non-autonomous* case with $f(t, y)$ again uses

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, \ldots, s \,, \tag{2.47}$$

see (2.29) in Remark 5 for explicit RK methods.

L-Stability Conditions

For the class of general RK methods, there exists a simple necessary condition for L-stability (recall (2.28) above), which we briefly want to derive here. Insertion of the Dahlquist test model $y' = \lambda y, y(0) = 1$ into the above RK formulas yields (in matrix-vector notation and with $e^T = (1, \ldots, 1)$)

$$y_1(z) = 1 + zb^T(I - z\mathcal{A})^{-1}e = 1 + b^T(\frac{1}{z}I - \mathcal{A})^{-1}e \,.$$

Obviously, if the matrix $\mathcal{A}$ can be assumed to be nonsingular, then one ends up with the result

$$y_1(\infty) = 1 - b^T\mathcal{A}^{-1}e \,. \tag{2.48}$$

This formula has an interesting consequence. Assume the vector b is equivalent to any row (say j) of the matrix $\mathcal{A}$, i.e. $b^T = e_j^T\mathcal{A}$, then one gets

$$y_1(\infty) = 1 - e_j^T\mathcal{A}\mathcal{A}^{-1}e = 1 - e_j^Te = 0 \,. \tag{2.49}$$

In other words: If the vector b is equivalent to any row of the nonsingular matrix $\mathcal{A}$ and the corresponding implicit RK method is A-stable, then it is L-stable. We will make use of this property below.

Collocation Approach

Rather than constructing implicit RK methods via solving the many algebraic equations for the coefficients, one may construct a subset of such methods, called *collocation methods*, by some direct approach. Therein the continuous solution $y(t)$ is approximated by some polynomial function $u(t)$ with

$$u(0) = y_0, \quad u(\tau) = y_1$$

that satisfies the following *collocation* conditions:

$$u'(c_i\tau) = f(u(c_i\tau)), \quad i = 1, \ldots, s \,. \tag{2.50}$$

The prescribed (normalized) collocation nodes

$$0 \le c_1 < \ldots < c_s \le 1$$

characterize the method. Assuming such a polynomial exists, we may introduce a Lagrange basis $\{L_1, \ldots, L_s\}$ with respect to the nodes c_i. If we identify

$$k_i = u'(c_i\tau), \quad i = 1, \ldots, s \,,$$

we may write the polynomial derivative as

$$u'(\theta\tau) = \sum_{j=1}^{s} k_j L_j(\theta) \,. \tag{2.51}$$

Upon integrating this relation, we obtain

$$u(c_i\tau) = y_0 + \tau \int_{\theta=0}^{c_i} u'(\theta\tau) d\theta = y_0 + \tau \sum_{j=1}^{s} a_{ij} k_j \,,$$

where we have defined

$$a_{ij} = \int_{\theta=0}^{c_i} L_j(\theta) d\theta, \quad i,j = 1, \ldots, s \,. \tag{2.52}$$

Insertion of these definitions into the collocation conditions (2.50) verifies that collocation methods are in fact special implicit RK methods. Moreover, we obtain

$$y_1 = y_0 + \tau \int_{\theta=0}^{1} u'(\theta\tau) d\theta = y_0 + \tau \sum_{j=1}^{s} b_j k_j \,,$$

where we have defined

$$b_j = \int_{\theta=0}^{1} L_j(\theta) d\theta, \quad j = 1, \ldots, s \,. \tag{2.53}$$

Thus we have all pieces of an implicit RK method together: With the choice of the (normalized) collocation nodes $c_1, \ldots, c_s$ we can define the Lagrange polynomials $L_1, \ldots, L_s$ and thus via (2.52) and (2.53) the coefficients $(b, \mathcal{A})$.

Discretization Error

The discrepancy between the ODE solution and the collocation polynomial is given by

$$y(\tau) - u(\tau) = \int_{\theta=0}^{1} f(y(\theta\tau))d\theta - \sum_{j=1}^{s} b_j f(u(c_j\tau)) = \mathcal{O}(\tau^{p+1}) \ . \tag{2.54}$$

Obviously, this represents some *quadrature* error (see, e.g., [17, Section 9.2]) of consistency order p.

In order to determine the maximally possible consistency order p, we arrive at the *Gauss-Legendre* quadrature, see, e.g., [17, Section 9.3]. In this algorithm, the collocation nodes are just the zeros of the *Legendre polynomial* $P_s(\theta), \theta \in [0, 1]$, which satisfy

$$0 < c_1 < \ldots < c_s < 1 \ .$$

The corresponding *Gauss collocation method* exhibits the following properties:

- Consistency order $p = 2s$ (maximum possible order).
- A-stability with $\mathcal{S} = \mathbb{C}_-$.
- Asymptotic behavior $|y_1(\infty)| = 1$.

This class of methods has further intriguing properties, which are discussed in [16, Section 6.3.4], but usually do not play a role in applications within systems biology. Here we would prefer an L-stable discretization method. That is why, in view of (2.49), a quadrature rule with $c_s = 1$ is selected, also named *Radau quadrature*. The remaining coefficients $c_1, \ldots, c_{s-1}$ are then the zeros of the *Jacobi polynomial* $P_{s-1}^{(0,1)}(\theta)$ so that

$$0 < c_1 < \ldots < c_{s-1} < 1 \ .$$

Under this sufficient assumption, the coefficient matrix $\mathcal{A}$ can be shown to be nonsingular (proof skipped here).

The thus constructed *Radau collocation method* is characterized by the following properties:

- consistency order $p = 2s - 1$,
- A-stability $\mathbb{C}_- \subset \mathcal{S}$,
- asymptotic stability $y_1(\infty) = 0$.

Hence, this collocation method is L-*stable*, as desired.

Radau Collocation Codes

A highly efficient code, named RADAU5, has been implemented by E. Hairer, see [33], and can be downloaded from his homepage. This code, obviously realizing the case $s = 3, p = 5$, uses a Newton-like iteration to solve the arising sd nonlinear equations. In each of these iterations, the direct solution of the linear equations is speeded up by factor of 5 exploiting the special structure of the arising matrix. Since the method is of collocation type, there exists a natural interpolation $u(\theta\tau)$ for all values $\theta \in [0, 1]$, which is the basis for a "dense output" option; in turn, this opens the door to a possible application to *delay* or *retarded differential equations*, see (1.5). The corresponding code RADAR5 due to N. Guglielmi and E. Hairer, see [31], is a direct extension of RADAU5 for this class of problems.

2.3.2 BDF Method

In this multistep approach, we start from the differential equation in its original form at time point $t_1 = \tau$, i.e.

$$y'(\tau) = f(y(\tau)) \tag{2.55}$$

and replace the unknown solution y by a polynomial g_k interpolating the $k + 1$ "previous" values

$$y_1, y_0, y_{-1}, \ldots, y_{-k+1}$$

on the uniform grid $t_1, t_0, t_{-1}, \ldots, t_{-k+1}$ with stepsize τ. This polynomial is uniquely defined so that we may write

$$y(t) \quad \rightarrow \quad g_k(t) = \sum_{j=0}^{k} L_j(t) y_{j-k+1}$$

where the L_j denote the Lagrange polynomials that depend on the interpolation nodes only, see, e.g., [17, Section 7.1]. Insertion of the expression

$$g_k'(\tau) = \sum_{j=0}^{k} L_j'(\tau) y_{j-k+1} = \frac{1}{\tau} \sum_{j=0}^{k} L_j'(0) y_{j-k+1}$$

into (2.55) yields the discretization scheme

$$\alpha_k y_1 + \ldots + \alpha_0 y_{-k+1} = \tau f(y_1) \tag{2.56}$$

in terms of uniquely defined coefficients $\alpha_j = L_j'(0)$. This special implicit multistep method has been suggested for stiff integration by C. W. Gear [28] in 1971. The relation (2.56) may be interpreted as an interpolation formula for numerical differentiation based on backward values, which gave the name *backward differentiation formula*, briefly: BDF.

In each step of a k-order BDF method, a system of only d nonlinear equations for y_1 has to be solved; note that the dimension of the system is independent of the order k. For $k = 1$ the implicit Euler method arises. The nonlinear systems must be solved by some Newton-like method, since any fixed point iteration would not be appropriate for stiff ODE problems. Consequently, several linear systems of the kind (ignoring the specific argument by merely writing $(\cdot)$)

$$\left(\alpha_k I_d - \tau f_y(\cdot)\right) \Delta y = \tau f(\cdot) \; .$$

must be solved until convergence of the Newton-like iteration.

Stability Properties

Upon inserting the usual Dahlquist test model (2.23), the discretization (2.56) yields

$$y_1(z) = -\frac{\alpha_{k-1} y_0 + \ldots + \alpha_0 y_{-k+1}}{\alpha_k - z} \; .$$

For $z \to \infty$ we thus obtain

$$y_1(\infty) = 0$$

which clearly looks like an extension of one of the conditions for L-stability of one-step methods. Of course, we would need A-stability in addition. However, stability analysis of multistep discretizations is much more subtle than for one-step methods; for the purpose of this book we do not dwell on the details. Important in our context is the so-called *second Dahlquist barrier* (see, e.g., [16, Section 7.2.2]): It states that *multistep methods with order $k > 2$ cannot be A-stable*, which rules out L-stability for $k > 2$. As the coefficients are determined, we must be content with L(α)-stability as it comes out, see Table 2.3. Not shown in the table is that the method is not even consistent for $k > 6$.

Due to the drastic deterioration of L(α)-stability for increasing order, *oscillatory phenomena* should be computed with order $k \leq 2$. There are well-known test examples with oscillatory behavior where BDF methods of higher order slow down

Table 2.3 L(α)-stability of BDF methods. Observe the second Dahlquist barrier

k	1	2	3	4	5	6
α	90°	90°	∼86°	∼73°	∼52°	∼18°

significantly or even produce wrong results, see [14]. Recall that $k = 1$ alone is not a really good idea, since the implicit Euler method is known to exhibit the unwanted "superstability".

Order and Step-Size Control

As for the Adams scheme, the BDF scheme, too, requires suitable starting values. Under certain (unrealistic) assumptions on such starting values, a rough examination supplies the discretization error

$$y_1 - y(\tau) = \mathcal{O}(\tau^{k+1}), \tag{2.57}$$

To start a scheme of order $\overline{k}$, one must implement a *start-up procedure*

$$k = 1, 2, \ldots, \overline{k} \leq 6 \, .$$

The starting step is always an implicit Euler step. With such a procedure, the above discretization error bound changes from step to step.

Just as in the Adams case, algorithmic difficulties in the *adaptive* selection of order and step sizes arise. The derivatives of the Lagrangian polynomials change whenever the local step sizes change, leading to some non-equidistant grid. This leads to the dilemma of implementation as either a Nordsieck variant or a divided difference variant. By and large, the method gains its best efficiency when run with constant step size and fixed order, which makes it a bit less robust than one-step stiff integrators.

BDF Codes

Among the most popular and efficient BDF codes are: the code LSODI (I stands for implicit) due to A. C. Hindmarsh [36]; the code LSODA (A stands for automatic switching) which automatically switches between the (implicit) BDF and the (explicit) Adams method; VODE due to [8], a variable step size/variable order BDF code, or the most recent code DASSL due to L. Petzold [51]. Due to their structure based on interpolation, a natural "dense output" option is usually available.

2.4 Linearly Implicit One-Step Methods

This section is devoted to efficient numerical *stiff integrators* that, in contrast to the methods of the previous Sect. 2.3, merely require the numerical solution of a low fixed number of *linear* equations per discretization step.

General Idea

The key issue in numerical stiff integration is the correct treatment of asymptotic stability of a given ODE model by step sizes that reflect the smoothness of the "slow" part of the solution, not that of the "fast" transition part of the solution, in cases where this is of less interest. In order to tackle this issue, one may subtract a linear homogeneous term on both sides of the ODE thus obtaining

$$\underbrace{y' - Jy}_{\text{implicit}} = \underbrace{f(y) - Jy}_{\text{explicit}} =: \bar{f}(y), \qquad y(0) = y_0 \ . \tag{2.58}$$

For stability reasons, the matrix J herein is either the Jacobian $J = f_y(y_0)$ or some approximation of it. The idea is to discretize the linear part on the left side implicitly (hence the name), but the "deflated" right-hand side $\bar{f}(y)$ explicitly. Clearly, such discretization schemes are computationally easier to realize than the fully implicit schemes from the preceding Sect. 2.3.

2.4.1 Rosenbrock-Wanner Methods

In 1963, H. H. Rosenbrock suggested a linearly implicit extension of explicit RK methods, which was later modified and improved by G. Wanner [33]. That is why such schemes today are called *Rosenbrock-Wanner* (ROW) schemes:

$$\left(I - \tau\beta_{ii} J\right)k_i \ = \ \tau \left[\sum_{j=1}^{i-1} (\beta_{ij} - \alpha_{ij})\, Jk_j + f\left(y_0 + \tau \sum_{j=1}^{i-1} a_{ij}k_j\right) \right] \tag{2.59}$$

The fact that the first right-hand sum above ends at index $i-1$ indicates that the system is block-triangular, while the second sum is anyway explicit. If the β_{ii} are all different, then a sequence of s linear systems with (d, d)-matrices $I - \tau\beta_{ii}J$ must be solved numerically. To simplify the linear algebra, very early the choice

$$\beta_{ii} = \beta, \ i = 1, \ldots, s$$

has been suggested, which implies that only one matrix $I - \tau\beta J$ needs to be decomposed throughout all stages.

In the above ROW methods, the identification $J = f_y(y_0)$ is strictly assumed. As in general RK methods, the coefficients $(\alpha_{ij}), (\beta_{ij})$ must be determined such that $N_p^{\mathrm{ROW}} = N_p^{\mathrm{RK}}$ algebraic equations corresponding to order p must be satisfied, see Table 2.4. If, however, the Jacobian J is replaced by an arbitrary Jacobian approximation matrix W, then one speaks of W-*methods*. As a consequence, a larger number N_p^{W} of algebraic equations needs to be satisfied, see again Table 2.4.

Table 2.4 Number of algebraic conditions to be satisfied by coefficients (α_{ij}), (β_{ij}) of ROW- versus W-methods

p	1	2	3	4	5	6	7
N_p^{ROW}	1	2	4	8	17	37	85
N_p^{W}	1	3	8	21	58	166	498

ROW Codes

As in Runge-Kutta methods, a whole "ROW-technology" has evolved over the years that led to a large number of implementations. In most cases, ROW methods of low order have been developed, for reasons clear from Table 2.4. Construction principles were, of course, the desirable L-stability and economy of evaluations as well as matrix decompositions and forward/backward substitutions. Among the most efficient codes of this kind are certainly

- ROS3PL with $p = 3$ and $s = 4$ due to J. Lang and D. Teleaga [43], an L-stable ROW-method, which is robust against Jacobian perturbations (even though not a full W-method),
- RODAS with $p = 4$ and $s = 6$ due to E. Hairer and G. Wanner [33]

All these codes are *adaptive*, which here means they possess an automatic step-size control, but keep the order fixed. A "dense output" option can be naturally realized within the embedding of the ROW methods.

2.4.2 Extrapolation Methods

As in the non-stiff case, the construction of a subset of linearly implicit Runge-Kutta methods can be directly realized via extrapolation, thus avoiding the cumbersome solution of the many algebraic equations for the coefficients. In such methods, one merely has to apply some well chosen basic discretization scheme of lowest order that is suitable for stiff integration. Higher orders are then obtained via the Aitken-Neville algorithm. Order and step-size control is realized just as in the explicit extrapolation methods. For ODE problems in systems biology, two kinds of schemes are useful.

Linearly Implicit Euler Discretization

Starting from the basic idea in (2.58), one discretizes the linear part on the left by the implicit Euler scheme, which leads to (for $n = 0, 1, \ldots$ and internal step size σ)

$$\eta_{n+1} = (I - \sigma J)^{-1} \left(\eta_n + \sigma \bar{f}(y_n)\right) = \eta_n + \sigma (I - \sigma J)^{-1} f(y_n) \ . \tag{2.60}$$

Table 2.5 Linearly implicit Euler scheme with extrapolation when subdivision $\mathcal{F}_H$ is chosen. L(α)-stability of subdiagonal elements $T_{k+1,k}$ in the extrapolation tableau

k	1	2	$3 \cdots 7$
α	90°	90°	$\geq$89.77°

Note that this is formally some W-method, since the matrix J is not required to be the exact Jacobian $f_y(y_n)$, but will usually be selected as some Jacobian approximation $J \approx f_y(y_0)$. In passing we note that for $J = 0$ we obtain the extrapolation method based on the explicit Euler scheme. This basic scheme is run repeatedly with successively smaller internal step sizes (compare Sect. 2.2.2 above)

$$\sigma_k = \tau/n_k, \quad n_k \in \mathcal{F}_H = \{1, 2, 3, 4, \ldots\} .$$

This implies that n_k linear systems need to be solved requiring the decomposition of (d, d)-matrices $I - \sigma_k J$ and the corresponding number of forward/backward substitutions.

From theory, one knows that this discretization permits an asymptotic σ-expansion, see [16, Section 6.4.2]. This is the theoretical basis for some σ-extrapolation, see Table 2.2 for the corresponding triangular Aitken-Neville scheme to compute elements T_{ik}.

Stability properties. Insertion of the Dahlquist test model (2.23) confirms that all elements T_{ik} (with $i \geq k$) of the extrapolation tableau satisfy the necessary condition (let $z = \lambda\tau$ and $J = \lambda$)

$$T_{ik}(z) \sim \frac{1}{z^{i-k+1}} \to 0 \quad \text{for} \quad z \to \infty .$$

This is one of the necessary conditions for L-stability. In Table 2.5, we arrange the L(α)-results for the subdiagonal elements in the extrapolation tableau, which are actually chosen for error as well as order and step-size control.

Dense output. As usual for adaptive extrapolation integrators, this one also selects rather large step sizes due to its efficient order and step-size control. A cubic Hermite interpolation tool has been constructed in [19] based on the information $y_0, f(y_0)$ and $y_1, f(y_1)$. In the DAE case, good approximations for $f(y_1)$ are obtained via σ-extrapolation based on the values $(\eta_n - \eta_{n-1})/\sigma$ for $n \in \mathcal{F}_H$. An efficient higher order technique has been worked out by E. Hairer and A. Ostermann [32].

Linearly Implicit Midpoint Rule

As in the non-stiff case, we would prefer to construct some basic discretization scheme that permits σ^2-extrapolation. This has been achieved by G. Bader and P. Deuflhard [3]. As a symmetric extension of the explicit mid-point rule, they introduced the *linearly implicit midpoint rule* according to

$$(I - \sigma J)\eta_{n+1} - (I + \sigma J)\eta_{n-1} = 2\sigma \bar{f}(y_n) \tag{2.61}$$

to be started by a linearly implicit Euler step. Instead of an extension of the Gragg final step (2.38), they introduced a different symmetric final step, which also requires an additional evaluation of $f(\eta_{n_\nu})$,

$$y_1(\tau;\sigma_\nu) := \hat{\eta}_{n_\nu} = \frac{1}{2}(\eta_{n_\nu+1} + \eta_{n_\nu-1}), \quad n_\nu \text{ even} . \tag{2.62}$$

The reason for this final step will be explained below in the context of stability.

Stability properties. Insertion of the Dahlquist test model (2.23) yields

- for the odd indices

$$\eta_{2m+1} = \frac{1}{1-z}\left(\frac{1+z}{1-z}\right)^{m-1} \;\to\; \frac{(-1)^m}{z} \;\to\; 0 , \tag{2.63}$$

- for the even indices

$$\eta_{2m} = \left(\frac{1+z}{1-z}\right)^{m} \;\to\; (-1)^m \;\to\; 0 , \tag{2.64}$$

- and for Bader's symmetric final step

$$\hat{\eta}_{2m} = \frac{1}{2}(\eta_{2m+1} + \eta_{2m-1}) \;\to\; \frac{(-1)^{m-1}}{z^2} \;\to\; 0 ,$$

  which can be seen to perform some asymptotic smoothing.

In order to have the same asymptotic sign pattern for all smoothing steps, one arrives at the sequence $\mathcal{F}_\alpha$ shown below. Clearly, with these specifications, the asymptotic result

$$T_{ik}(z) \sim \frac{-1}{z^2} \to 0 \quad \text{for} \quad z \to \infty$$

is obtained, i.e. one has a uniform asymptotic pattern throughout the whole extrapolation table. We are thus only left to study the L(α)-stability pattern, which is given in Table 2.6.

Table 2.6 Linearly implicit mid-point rule with extrapolation when subdivision sequence $\mathcal{F}_\alpha$ is chosen. L(α)-stability of subdiagonal elements $T_{k+1,k}$ in the extrapolation table

k	1	2	3	4	5	6
p	1	3	5	7	9	11
α	90°	90°	∼88°	∼86°	∼87°	∼87°

Obviously, the stability properties including the final step are very satisfactory. However, from the detailed study of the intermediate discretization steps we learned that the asymptotic behavior (2.63) for the odd indices is satisfactory, while the behavior (2.64) for the even indices is unsatisfactory. As a consequence, this extrapolation method is recommendable mainly for "moderately stiff" problems, which, however, represent the typical problems in systems biology.

Quadratic extrapolation. For a scheme as specified, the existence of an asymptotic σ^2-expansion has been shown in [3]; it gives rise to a σ^2-extrapolation for *even* subdivision sequences. Additional conditions come from the above stability analysis that lead to the sequence

$$\sigma_k = \tau/n_k, \quad n_k \in \mathcal{F}_\alpha = \{2, 6, 10, 14, 22, 34, 50\}\,.$$

The index α comes from the empirically introduced property $n_{k+1}/n_k \le \alpha := 1.4$. These data enter into a triangular Aitken-Neville scheme with T_{ik} for $i \ge k$ as shown in Table 2.2. Needless to say that, of course, an adaptive version is implemented, see (2.41) for the step size control and (2.42) for the order control, the latter requiring some subtle decision about what should be inserted as computational work per discretization step.

Dense output. As in the other extrapolation integrators, a cubic Hermite interpolation polynomial can be constructed based on the information $y_0, f(y_0)$ and $y_1, f(y_1)$. The idea is the same as for the explicit midpoint rule: at the end of each τ-step one evaluates $f(y_1)$, which then can be recycled as $f(y_0)$ in the next integration step so that no additional function evaluation is needed (apart from the very last step). Again we mention that the accuracy of this Hermite interpolation formula in combination with the order and step-size control is enough for the purpose of systems biology.

Linearly Implicit Extrapolation Codes

The *adaptive* extrapolation code EULSIM has been designed on the basis of the linearly implicit Euler scheme (originally called S*emi-* IM*plicit* Euler scheme, hence the name). The more elaborate extrapolation code LIMEX due to [19, 23] is an extension that also applies to quasilinear differential-algebraic equations. For "moderately stiff" ODE problems, which in systems biology are the most frequent

case, the code METAN1 [3], based on the linearly implicit midpoint rule, typically supersedes LIMEX. For an illustration, see Sect. 2.5.2. In both codes, dense output options are available based on cubic Hermite interpolation. The code METAN1 also found its way into the book on "Numerical Recipes" [52], p. 735.

2.5 Choice of Numerical Integrator

A code may fail; but it must not lie.
(Beresford N. Parlett)

In the sections above, a number of different methods including efficient codes have been discussed to necessary detail. The present section is devoted to questions that a user may have when deciding which of these codes to apply for his problem at hand. In Sect. 2.5.1, we arrange methods and associated codes (written as CODES) again; for download addresses see the final chapter *Software* at the end of the book. In Sect. 2.5.2, we illustrate the choice of integrator at two moderate size examples from systems biology, which look rather similar in terms of the differential equation model, but behave differently in terms of the numerics. Finally, in Sect. 2.5.3, we present a quite challenging large scale problem dealing with the so-called Warburg effect of tumor cells.

2.5.1 A General Roadmap for Numerical Integrators

One-Step Methods

In the sections above we have discussed *explicit* one-step methods for *non-stiff* ODE problems such as

- explicit Runge-Kutta methods in Sect. 2.2.1 (DOPRI5 , DOP853), and
- extrapolation methods in Sect. 2.2.2 (DIFEX1, ODEX) ,

as well as *(linearly) implicit* methods for *stiff* problems such as

- Radau collocation methods in Sect. 2.3.1 (RADAU5, RADAR5),
- Rosenbrock-Wanner methods in Sect. 2.4.1 (ROS3PL, RODAS), and
- extrapolation methods in Sect. 2.4.2 (LIMEX, METAN1).

All of these codes select a "locally optimal" step size on the basis of local discretization error estimates, extrapolation methods also a "locally optimal" order, codes like DOP853 an optimal order among the three orders 8,5,3. In parallel with the dynamics of the ODE system, *non-uniform* grids are obtained with *problem dependent* output points, typically much less than with uniform grids. If more than the automatically computed output data are wanted, which is often called the "dense" output option, then extra tools for interpolation are appropriate to avoid

wasting computing time due to "too many" output points; such extra tools are easily available in embedded RK methods as well as in collocation methods, while extrapolation methods require an additional (computationally cheap) device based on Hermite interpolation. Compared to multistep methods, implementations of these methods exhibit only a small amount of overhead beyond f-evaluations. Generally speaking, these methods are particularly efficient in ODE problems with strongly varying dynamics, a feature that is especially true for extrapolation methods and DOP853, since they additionally choose some "locally optimal" order.

Multistep Methods

From the class of multistep methods we have discussed

- (*explicit*) Adams methods for *non-stiff* ODE problems in Sect. 2.2.3 (LSODE, DEABM),
- (*implicit*) BDF methods for *stiff* problems in Sect. 2.3.2 (LSODI, VODE, and DASSL), and
- a multistep code that *automatically switches* between Adams and BDF method (LSODA).

All of these codes realize some control of order and step size, but in a much more restricted sense than in one-step methods. Efficient implementations for a change of *order* are different from those for a change of *step sizes*. Generally speaking, multistep methods gain their efficiency with *quasi-uniform* grids, a property prohibitive for problems with strongly varying dynamics, but in favor of smoothly varying dynamics; this goes with the intuition that a smooth dynamics can gain efficiency from exploiting the "history" of the trajectory. For smooth dynamics, the number of evaluations of the right-hand side f may be considerably less compared to one-step methods. For the BDF method, an order restriction $k \leq 2$ is recommended when applied to oscillatory problems, which do occur in systems biological networks. By their common construction principle via interpolation, both Adams and BDF methods naturally generate "dense" output, see Sect. 2.2.1. In comparison with one-step methods, they usually require much more computational overhead which often outweighs the possibly lower number of f-evaluations.

Non-stiff Versus Stiff Integration

The question of whether a given ODE problem should be regarded as stiff or non-stiff, stands at the beginning of each systems biological simulation. For a non-stiff problem, an explicit method will do, which only requires evaluations of the right-hand sides f and thus is faster per integration step than an implicit method. For a stiff problem, implicit or a linearly implicit methods may pay off, which additionally require the numerical solution of linear equations involving the Jacobian of the right-hand side, but over significantly less integration points. For really stiff problems, an

explicit method would suffer from severe *step-size restrictions* that would blow up the overall computing time.

Qualitative insight. As for the practical classification "stiff/non-stiff" in a given ODE problem, quite often some qualitative insight is helpful before starting the simulation: stiff problems are typically characterized by the fact that they asymptotically approach some *steady state point*. Then, generally speaking, an implicit or linearly implicit method should be applied. However, even for such problems, an explicit method might be preferable, if one is only interested in the "transition" phase.

Rule of thumb. As for the theoretical distinction between stiff and non-stiff ODEs, we have elaborated quite a bit on this question, but finally recommended a rather pragmatic approach: ODE problems, wherein the extra computational amount required for the arising nonlinear (or linear) equations pays off, are regarded as stiff. Consequently, whenever a stiff/non-stiff characterization of an ODE problem is unclear, the following rule of thumb is advised: start with an explicit method, say DOPRI5, and only switch to an implicit or linearly implicit method, say METAN1, if the explicit method seems to suffer from step-size restrictions that seem "uninterpretable" in view of the underlying model.

For an illustration of stiff versus non-stiff ODE problems, the following example may serve.

Example 8 (Gene expression) Let a gene expression be described by the scheme

$$\text{gene} \xrightarrow{k_1} \text{mRNA} \xrightarrow{d_1} \emptyset,$$

$$\emptyset \xrightarrow{k_2 \cdot [\text{mRNA}]} \text{protein} \xrightarrow{d_2} \emptyset,$$

wherein k_1 is the constitutive transcription rate, k_2 the translation rate, d_1 the mRNA degradation rate, and d_2 the protein degradation rate. We assume that this gene expression is unregulated, i.e. the gene is always on, which can be modelled by setting its concentration $g = [\text{gene}] \equiv 1$. Let $m = [\text{mRNA}]$ the concentration of mRNA and $p = [\text{protein}]$ the concentration of the protein. Thus one arrives at the ODE initial value problem

$$m' = k_1 - d_1 m,\ m(0) = 1, \quad p' = k_2 m - d_2 p,\ p(0) = 0\ .$$

Let the kinetic parameters be selected as $k_1 = 2$, $d_1 = 1$, $k_2 = 1$, $d_2 = 0.01$. The steady state point is $m^* = k_1/d_1$, $p^* = (k_1 k_2)/(d_1 d_2)$. The numerical solution is shown in Fig. 2.10. As can be observed, mRNA reaches its steady state much faster than the protein. This implies that the problem can be regarded as stiff. In Fig. 2.11, the step sizes chosen by a non-stiff integrator (here: DOPRI5) are compared with those chosen by a stiff integrator (here: LIMEX).

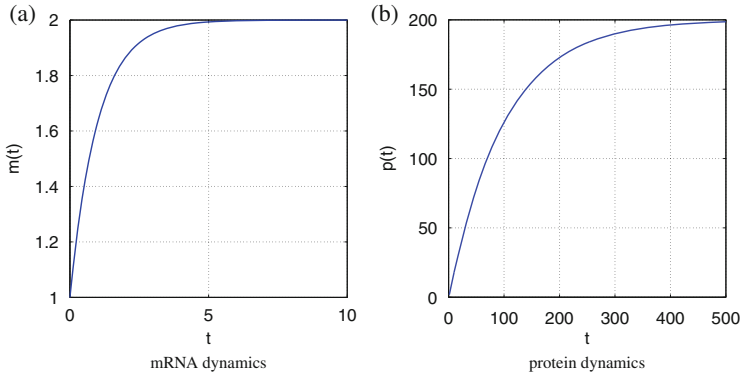


Fig. 2.10 Numerical solution for Example 8. Note the different time-scales in the two plots: The mRNA concentrations changes much more rapidly than the protein concentration

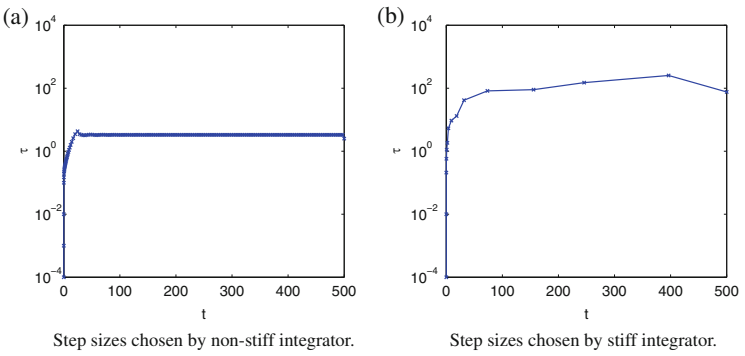


Fig. 2.11 Example 8. Automatic step-size selection by two different integrators. Common local error tolerance TOL$= 10^{-6}$. *Left*: Non-stiff integrator DOPRI5. "Unreasonable" step-size restriction with many ups and downs in the steady state phase. *Right*: Stiff integrator LIMEX. No step-size restriction in the steady state phase

Remark 8 For quite a while research has focused on the construction of methods that *automatically* classify stiff versus non-stiff problems, switching between explicit and implicit codes during the computation. An example of this type is the quite popular multistep code LSODA that automatically switches between stiff (BDF) and non-stiff (Adams) methods. There is, however, a principal difficulty: The distinction is easy in an implicit method (here: BDF), since there Jacobian information is available; but then, in a non-stiff problem, the bulk of the computational amount has already been spent so that not too much computing time can be saved. The distinction is difficult in an explicit method (here: Adams), since there the necessary information on the Jacobian matrix is missing.

Computational Speed

Suppose the decision between non-stiff or stiff integrator has been made. Then the choice among explicit or implicit integrators, respectively, must be made. The task to find the "fastest" code for the problem at hand is not as easy as one might expect. In published comparisons of computing times for different codes, the multistep community quite often only gives the number of f-evaluations excluding overhead, while the one-step community tends to present only total computing times, which often depend on the selected computer.

Non-stiff integrators. For this class of methods, the total computing time (CPU time) originates from the number N_f of f-evaluations (at a cost of C_f each) and the overhead $\Omega = \omega_{\text{method}} \cdot d$, usually proportional to the dimension d of the ODE system with a method dependent proportionality factor ω_{method}. Thus we arrive at

$$\text{CPU} = C_f \cdot N_f + \Omega = C_f \cdot N_f + \omega_{\text{method}} \cdot d \ .$$

As explained above, multistep (ms) and one-step (os) methods may be characterized by the relations

$$N_f^{\text{ms}} \leq N_f^{\text{os}}, \quad \omega_{\text{os}} \ll \omega_{\text{ms}} \ .$$

Obviously, the distinguishing quantity will be

$$\gamma = C_f/d \ ,$$

which can be interpreted as "evaluation time per component of the right-hand side f". From it, we may derive the following *rule of thumb*:

- Whenever γ is "not too large", then some explicit Runge-Kutta or extrapolation method should be chosen; in systems biology, this seems to be the most frequently occurring case, since each component typically couples only with few other components.
- If γ is "large", then one should prefer an Adams method.

If the expected dynamics is "strongly varying", then a one-step method should be taken anyway. Compared to the optimized explicit Runge-Kutta codes DOPRI5 and DOP853, the two extrapolation codes DIFEX1 or ODEX typically are regarded as slightly slower in standard non-stiff ODE problems, but slightly more robust in challenging real life problems.

Stiff integrators. For implicit and linearly implicit integrators there is no such *simple* complexity theory as in the non-stiff case. Here we have to additionally count the number of matrix decompositions and of forward/backward substitutions. This confuses the picture quite a bit. What remains valid is that the BDF method as a multistep method also requires a much larger overhead than the one-step competitors. Moreover, the linearly implicit one-step methods (Rosenbrock-Wanner or extrapolation method) are much simpler than their implicit counterparts, which

particularly pays off for large ODE systems, when the arising linear systems may even be solved iteratively.

Accuracy

Recall from Sects. 2.1.1 and 2.1.2 that a user of a numerical integrator can only prescribe a local error tolerance TOL, typically split into RTOL, a relative error tolerance, and ATOL, an absolute error tolerance. Note that RTOL relies on an efficient scaling (see the associated item below). The achieved global accuracy ERR then depends on the condition of the problem. In real life problems, the condition number is usually computationally unavailable. Hence, a user should develop some "feeling" about the necessary error tolerance. As for the choice of the integrator, the achieved accuracy will mostly be better, if less integration points are needed – a feature that usually speaks for extrapolation methods.

Computational Parameter Sensitivity Analysis

In the class of problems envisioned here, the additional numerical integration of the sensitivity equations will usually come up. From Sect. 1.3.2, we recall that the sensitivity y_p with respect to some parameter p (dropping the index) is described by the equations

$$y_p' = f_y(y(t), p)y_p + f_p, \quad y_p(0) = 0 \ .$$

As already mentioned in Remark 4, these equations must be solved simultaneously with the original model equations $y' = f(y)$ to obtain the argument inside $f_y(\cdot), f_p(\cdot)$. The most convenient way to realize the above variational equation is to generate the exact formulas for f_y and f_p from some chemical compiler simultaneously with the generation of the right hand side f (see Sect. 1.2.3).

If some *stiff* integrator is employed, then the Jacobian f_y and the decomposition of the matrix $I - \beta\tau f_y(y_0)$ can be also included in the integration of the state variable ODEs. Within any *linearly* implicit one-step method, the same idea as in (2.58) works again in the form

$$\underbrace{y_p' - Jy_p}_{\text{implicit}} = \underbrace{(f_y(y,p) - J)y_p + f_p}_{\text{explicit}} \qquad y_p(0) = 0 \ . \tag{2.65}$$

An efficient implementation of this idea within the code LIMEX has been suggested and worked out by M. Schlegel et al. [55], which pays off especially in large ODE networks. For BDF methods, which require the iterative solution of *nonlinear* equations, an especially adapted technique has been worked out by T. Maly and L. Petzold [44].

Discontinuity Treatment

In some applications, the right-hand sides f contain discontinuities at certain points. In systems biology, such a situation typically occurs when different models are used to describe different processes before or after some characteristic event (day-night, say). If one ignores such points by just "overriding" them with a numerical integrator, then usually the accuracy after these points will be poor. This phenomenon occurs less marked, if the employed integrator is equipped with an automatic order control that allows for sudden local drops of order (as in extrapolation methods). In principle, however, this kind of difficulty should be tackled differently: The integration should be terminated at these points and restarted thereafter; in this way, the accuracy can be preserved. Note that this procedure is a structural disadvantage for any multistep method, which requires a restart from order $k = 1$ up to the locally optimal order after the discontinuity point. For this reason, certain one-step procedures have been designed to realize some "quick start-up".

Example 9 For the purpose of illustration, we consider an artificial example constructed by R.D. Russell and L.F. Shampine [54]:

$$y'' = y - ty' + te^t - |t|(6 - 12t + 2t^2 - 3t^3),$$

$$y(-1) = e^{-1} - 2, \quad y'(-1) = e^{-1} + 7.$$

Here a discontinuity of y''' occurs at $t = 0$, known in advance. The unique solution is

$$y(t) = \begin{cases} e^t + t^3 - t^4, & t \le 0 \\ e^t - t^3 + t^4, & t \ge 0 \end{cases}.$$

After the point $t = 0$, accuracy will be reduced, if a numerical integrator without order control is used, see Fig. 2.12. On the contrary, stop and restart of the integration at $t = 0$ preserves the accuracy also beyond $t = 0$.

Dense Output

Such an option treats the case that more output points are wanted than automatically delivered by the step-size (and possibly order) control. This situation may well occur in systems biological modelling, there mainly for print-out. In addition, as worked out in the next Chap. 3, this option may also be important for parameter identification, when measurements are "too dense" compared with the step sizes selected by the adaptive integrators. Suppose one stopped the integrator at each of these points, then "too much" computational effort would be wasted. In particular, integrators with adaptive order control would find the lowest possible orders as optimal, if the distance between two neighboring points were "too small". A dense output option

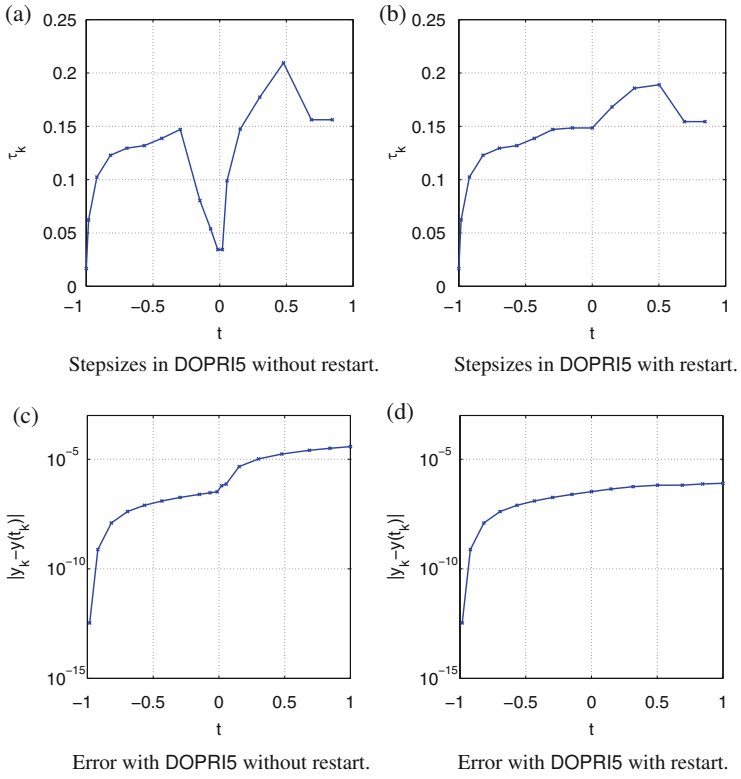


Fig. 2.12 Example 9. *Left:* A discontinuity of y''' at $t = 0$ causes step size reduction and accuracy loss. *Right:* Stop and restart of the integration at $t = 0$ preserves the accuracy also beyond $t = 0$

will usually leave the step-size control untouched and apply some interpolation formula for the points in between the automatically selected points. Such an option is easily implemented in multistep methods, Adams (Sect. 2.2.3) as well as BDF (Sect. 2.3.2) methods, since they are both constructed via interpolation formulas. In explicit RK methods, this option is conveniently realized via embedding techniques (Sect. 2.2.1). In the implicit Radau methods, there is anyway a natural representation of the whole trajectory via collocation. Extrapolation methods need to implement an extra device based on cubic Hermite interpolation; higher order dense output formulas have been worked out by E. Hairer and A. Ostermann [32].

Delay Differential Equations

As already mentioned in (1.5) in the introduction, the standard ODE system is sometimes replaced by a delay system. In biological systems, the *retardation* or

delay time $\tau > 0$ typically depends nonlinearly on the solution y, so that instead of (1.5) one should better write

$$y' = f\left(y(t), y(t - \tau(y(t)))\right), \quad y(t) = \Theta(t) \text{ for } [-\tau, 0]$$

with a given *initial function* Θ. Numerical integrators with a dense output option also permit the treatment of *delay* or *retarded differential equations* (DDEs), see (1.5). Among the explicit RK methods, we mention the code RETARD due to E. Hairer, among the implicit RK methods the code RADAR5 due to N. Guglielmi and E. Hairer [31]. In MATLAB, the solver dde23 due to L. F. Shampine [56] is provided for DDEs with *constant* delays, a case rare in systems biology; this code tracks discontinuities and integrates numerically by the explicit Runge-Kutta (2,3) pair and an interpolant implemented within MATLAB's ode23.

Reliability

A numerical integrator is said to be "reliable", if the delivered solutions are "accurate enough" compared with the condition of the problem at hand. For a basic understanding of this issue recall Sects. 2.1.1 and 2.1.2 where the relation of local and global accuracy has been discussed in some detail. From this discussion we know that *local* discretization errors need to be estimated; these estimates then enter into some adaptive control of step sizes (and possibly also orders). The finally achieved accuracy additionally depends on the structure of the underlying ODE system and the number of integration points. In this respect, one-step methods, in particular extrapolation methods, have a natural advantage, since they require less integration points than multistep methods.

Robustness

In the world of mathematical ODE modelling, this is the most important property: a numerical integrator should solve a large class of given problems (no matter how difficult) without much ado. Apart from the many details discussed above, robustness typically requires additional heuristic strategies (e.g., avoid division by zero, find a reasonable starting step size) and a careful implementation of software.

Scaling

Robustness typically requires a subtle application of scaling techniques within the code, an intricate issue that we have not touched upon in this book. Apart from any *external* scaling of variables prescribed by the user, robust codes often realize some *internal* scaling. Such a device is necessary to assure that any relative versus absolute error criterion works. Moreover, quite often "small" elements of the

numerical sensitivity matrices are set to zero – a device only reasonable when the term "small" is defined, which, in turn, is only reasonable, if scaled quantities are treated.

2.5.2 *Different Numerical Behavior in Two Similar Problems*

In this section, we present two problems from systems biology that, at first glance, look similar from the point of view of mathematical modelling. Both of them are of moderate size. They may serve as typical examples for how to deal with larger problems.

Human Menstrual Cycle Problem GynCycle

This model, published in detail in [53], has already been presented above as Example 1. In Fig. 1.6, the *compartments* of the model have been illustrated. In Fig. 1.7, part of the corresponding *chemical model* has been presented in the usual form of a reaction diagram. Thus one arrives at a *mathematical model* with $d = 33$ ODEs and 114 parameters, from which 63 degrees of freedom could be identified by methods described in the subsequent Chap. 3.

Bovine Estrous Cycle Problem BovCycle

This model has been inspired by the above human menstrual cycle model. In fact, the endocrine mechanisms that regulate the bovine estrous cycle are rather similar to those of the human menstrual cycle. A first version has been published in [7]. Our subsequently presented computations refer to the more recent version [58], where further details can be found. As for the selected *compartments* for the physiological description, Fig. 1.6 can again serve as defining the terms. A subdiagram of the *chemical model* is given in Fig. 2.13, to be compared with the more elaborate human model in Fig. 1.7. Finally, a mathematical model with $d = 15$ ODEs and 60 parameters comes up.

Comparative Performance of Numerical Integrators

In Figs. 2.14 and 2.15, we show the comparative performance of several numerical integrators. The documented local error tolerances TOL, see (2.14), range within $10^{-3}, \ldots, 10^{-6}$, which is a reasonable range for typical problems from systems biology. As for the performance, we study both the comparative CPU times required on a Fujitsu Siemens Lifebook E8210 and the achieved global accuracies ERR, see (2.15). The CPU time is a reasonable performance measure, whenever both

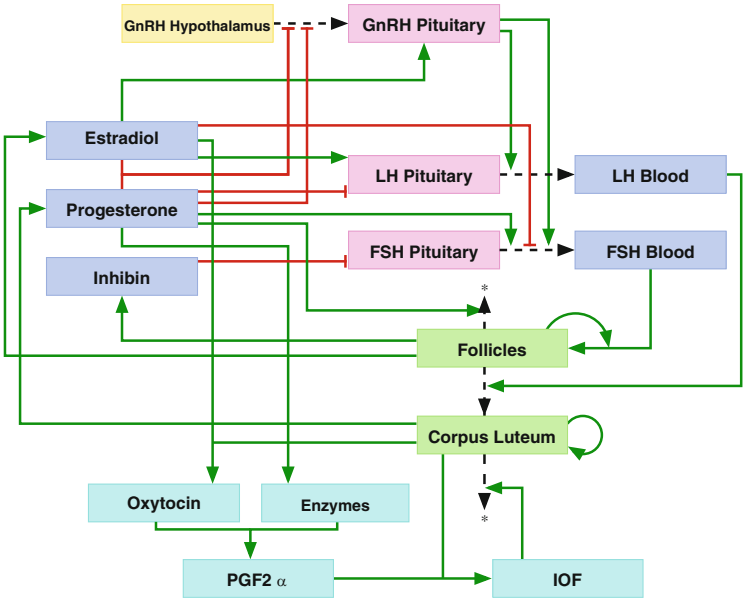


Fig. 2.13 Flowchart of a model for the bovine estrous cycle. *dashed lines*: transitions, elimination, or chemical reactions, *solid lines with filled arrows*: stimulatory effects (Hill functions h^+), *solid lines with unfilled arrows*: inhibitory mechanisms (Hill functions h^-)

non-stiff and stiff integrators are compared. As for the norm used to define ERR, we selected the *scaled root mean square error*, i.e.

$$\mathrm{ERR} = \left(\frac{1}{d} \sum_{i=1}^{d} \frac{(y_i(T) - y_{i,\mathrm{ref}}(T))^2}{y_{i,\mathrm{scal}}(T)} \right)^{1/2}, \tag{2.66}$$

wherein y_{ref} is the (highly accurate) computational result obtained with the extrapolation code LIMEX for TOL = 10^{-12} and y_{scal} is the scaling vector obtained during the computation.

Small test set of integrators. First, in order come to a fast decision about which numerical integrator to use, we test on a small subset of integrators that includes both stiff and non-stiff ones. On the basis of what has been presented in this chapter, let us select the non-stiff Runge-Kutta integrator DOPRI5, the stiff extrapolation integrator METAN1, and the mixed multistep code LSODA with automatic switching between a non-stiff Adams method and a stiff BDF method. The comparative results are presented in Fig. 2.15. From these numbers, we may gain the following insight:

- Problem BovCycle is non-stiff, as can be seen from the small amount of computing time of DOPRI5 versus METAN1.

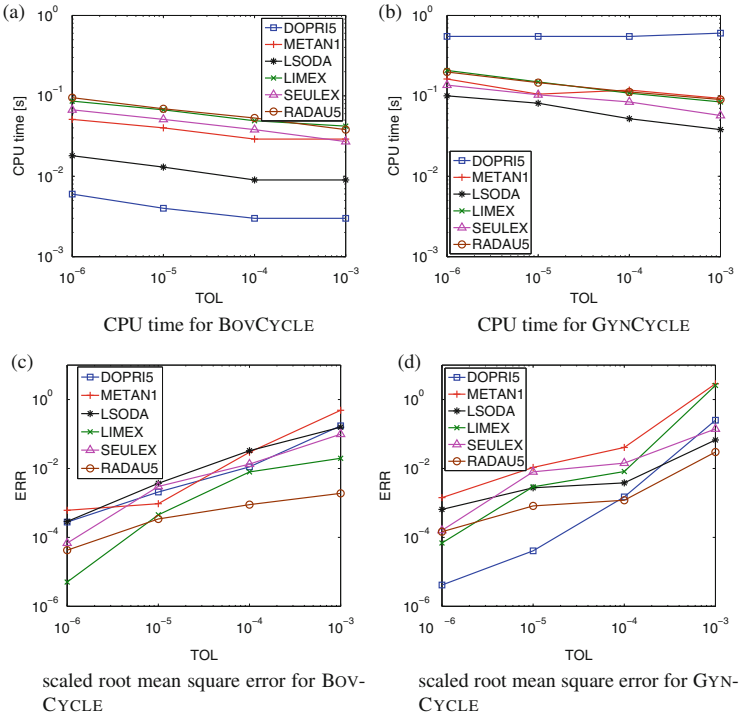


Fig. 2.14 Performance of six numerical integrators in two similar problems, to be compared with Fig. 2.15. *Left*: Bovine estrous cycle problem BovCycle. *Right*: Human menstrual cycle problem GynCycle. *Top*: CPU time. *Bottom*: Achieved global accuracy

- Vice versa, problem GynCycle is stiff, just compare the higher CPU of DOPRI5 versus METAN1.
- In the non-stiff problem, the accuracies of all three integrators are nearly the same.
- In the stiff problem, the accuracies spread by a factor of roughly 10, with the non-stiff integrator DOPRI5 surprisingly best, which yields to the insight that this problem is only *mildly* stiff.

The *number of steps* selected by the automatic step-size controls (not presented in detail here) is

- for BovCycle: between 10^3 with LSODA, DOPRI5 and 10^2 with METAN1,
- for GynCycle: between 10^4 with DOPRI5 and again 10^2 with METAN1.

In summary, Fig. 2.15 leads to the suggestion of using DOPRI5 in the non-stiff problem BovCycle, but LSODA or METAN1 in the stiff problem GynCycle.

Larger test set of integrators. Sometimes, users want a comparison over a larger test set of integrators. Therefore, beyond the three integrators DOPRI5,

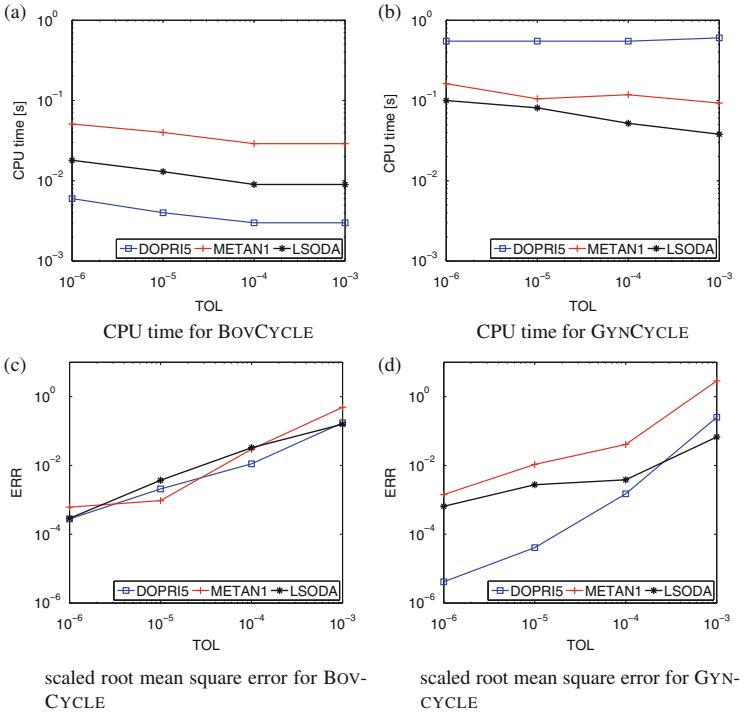


Fig. 2.15 Comparative performance of three numerical integrators in two similar problems. *Left*: Bovine estrous cycle problem BOVCYCLE. *Right*: Human menstrual cycle problem GYNCYCLE. *Top*: CPU time. *Bottom*: Achieved global accuracy

METAN1, LSODA above, we additionally include the three stiff integrators LIMEX, SEULEX, RADAU5. The comparative results are given in Fig. 2.14. From this larger data set, we conclude that the characterization "stiff versus non-stiff" remains the same. Moreover, the data about the computing times and achieved accuracies also remain essentially the same.

Remark 9 If, beyond the mere trajectory simulation, *sensitivity analysis* is wanted, then the linearly implicit extrapolation codes LIMEX, METAN1 have a structural advantage that also pays off as a gain in CPU time.

2.5.3 *Example: Warburg Effect in Tumor Cells*

This rather complex systems biological network has been worked out by M. König, H.-G. Holzhütter, and N. Berndt [42] from Charité, Berlin. We here partially follow their presentation. However, we focus on details of their numerical modeling, which is the topic of this monograph. Readers interested in more biological details are

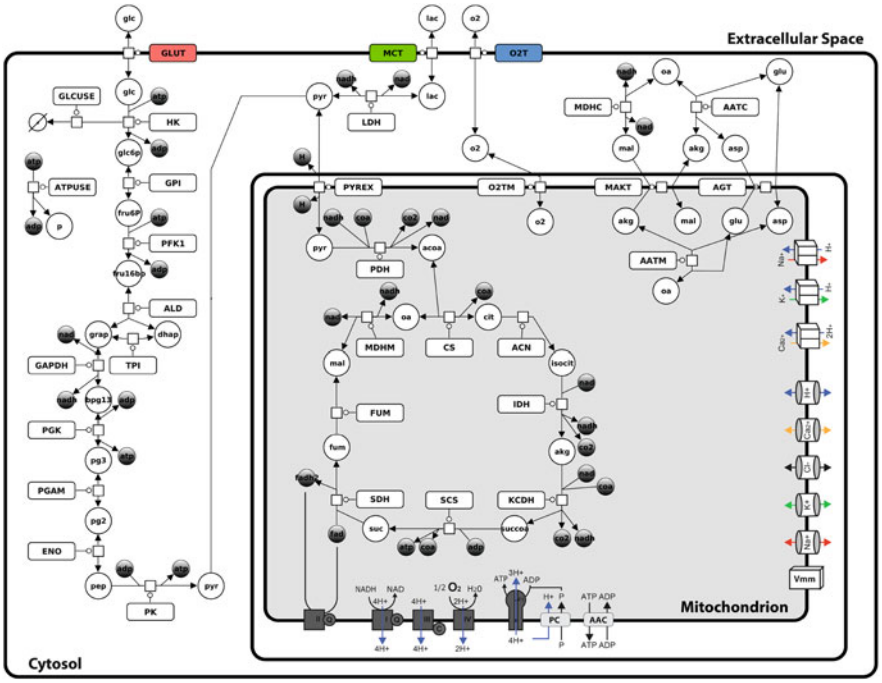


Fig. 2.16 Schematic representation of cellular-scale model for energy metabolism (Courtesy M. König)

referred to the original paper. The so-called *Warburg effect* means that solid tumor cells, as opposed to normal cells, exhibit an extraordinarily high demand for glucose even under aerobic conditions (i.e. in the presence of oxygen) with a substantial part of glucose being converted into lactate. In their work, the authors address the problem of whether zonation of the energy metabolism within a non-vascular tumor could serve as a means to influence its growth capacity.

Two-Scale Modeling

The authors of [42] developed an elaborate model for inter- as well as intracellular energy metabolism. Each cell is modeled via three compartments, the mitochondrion, the cytosol, and the extracellular space. Within these compartments, a kinetic model for 78 intracellular metabolites is realized. Figure 2.16 gives an impression of the complex metabolism within a single tumor cell. The most important processes are the tricarboxylic acid (TCA) cycle and the two central ATP delivering pathways, i.e. the glycolytic (GLY) pathway and the oxidative phosphorylation (OXP) pathway.

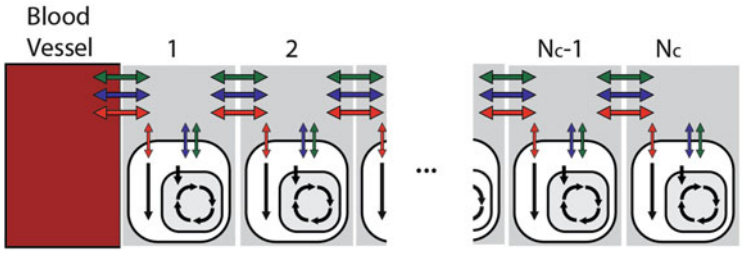


Fig. 2.17 Schematic representation of tissue-scale model of tumor metabolism. Note that Fig. 2.16 is a zoom into Fig. 2.17 (Courtesy M. König)

The coupling between tumor cells and with their tissue environment is again schematically described by a *compartment model*, see Fig. 2.17. The tumor as a whole is supplied with nutrients and oxygen via the nearest blood vessel (on the left of the figure). Exchangeable metabolites between the cells and the extracellular space are glucose, lactate and oxygen, which are assumed to diffuse between adjacent spatial layers – represented by double arrows in Fig. 2.17. In order to model the continuous diffusion process within the discrete compartment setting, a second-order finite difference approximation is used to replace the one-dimensional diffusion equation (which would be a partial differential equation).

Numerical Simulation

The total model above was specified to contain a fixed number of 25 cells, 5 extracellular compartments per cell, 78 intracellular metabolites per cell, and 3 extracellular metabolites. These model equations have been carefully programmed by hand to yield a system of 2328 ODEs. The project required repeated simulation runs within a *parameter study* that involved external oxygen and glucose availability or different metabolic strategies of energy production. This study required about 1000 simulation runs of the 2328 ODEs. Hence, computing time really matters here.

After first experiences of the authors with non-stiff integration, it quickly became clear that the ODE system is stiff (mainly due to the equations for the respiratory chain). First simulations with a stiff ODE code provided within a well-known commercial package could only be performed for up to 5 cells, which already required several minutes per run (to be multiplied by a factor of 1000 in the parameter study!). Simulations for the large systems just failed due to excess of storage requirement. The computational bottleneck turned out to be the solution of the large linear systems at each time step. In order to cope with this difficulty, the equations in the ODE system were reordered resulting in a banded structure of the Jacobian matrix: Metabolites and compartments related to individual cells were arranged within one block, as opposed to the original implementation that distributed elements over the whole system matrix. As a consequence, a numerical

band solver could be applied to solve the linear equations. Upon combining the two FORTRAN codes LIMEX for stiff integration (see Sect. 2.4.2) and MUMPS, a direct parallel sparse solver (due to [1, 2], here in band mode), the ODE system could be integrated within around 10 sec. This brought the whole parameter study within a tolerable region of computing time.

After the work reported in [42], the authors further developed their model so that today the ODE system is about three times larger than the one in the original publication and heavily relies on parameter studies. In order to further speed up computations, the present even larger system is integrated with the package RoadRunner, a .NET library for carrying out numerical simulations directly from given SBML models. RoadRunner uses the (implicit) BDF integrator CVODE [37, 38] (written in C) for differential equation solving and event handling. With this change, the simulation times of the larger ODE system could be reduced to about 3 sec. per run. Moreover, a modified scientific analysis now requires about 10.000 runs per parameter study.

If, however, sensitivity studies should be included, then the linearly implicit extrapolation codes like LIMEX or METAN1 (see Sect. 2.4.2) would again enter the game.

Chapter 3
Parameter Identification in ODE Models

In most systems biological models, a series of parameters enters that need to be discussed. In the preceding Sect. 2.5.3, we presented an example, where simulations for a large set of parameters have been performed and analyzed. This is the situation, when no measurements are available. The present chapter deals with the case, when measurements are available and can be used, in principle, to identify at least part of the parameters. Such parameter identification problems in ODE models typically arise as nonlinear least squares problems, see Sect. 3.1. They are solved by Gauss-Newton methods, which require the numerical solution of linear least squares problems within each iteration. For pedagogical reasons, the order of these three topics is reversed in our presentation. Therefore, in Sect. 3.2, linear least squares problems are discussed first including the important issue of automatic detection of rank deficiencies in matrix factorization. Clearly, not all data sets are equally well suited to fit all unknown parameters of a given model. Next, in Sect. 3.3, the class of "adequate" nonlinear least squares problems is defined, both theoretically and computationally, for which the local Gauss-Newton method converges. Globalization via a damping strategy is presented. The case of possible non-convergence is treated in detail to find out which part originates from an insufficient model and which one from "bad" initial guesses for the Gauss-Newton iteration. In Sect. 3.4, all pieces of the text presented so far are glued together to apply to the ODE models, which are the general topic of the book. Finally, in Sect. 3.5, three examples with increasing complexity are presented. First, the notorious predator-prey problem is revisited, which turns out to be quite standard. Next, in order to connect the advocated computational ideas with modelling intuition, a simple illustrative example is worked out in algorithmic detail. Last, a more complex parameter identification problem related to a model of the human menstrual cycle is discussed in detail.

In order to connect the advocated computational ideas with modelling intuition, a simple illustrative example is worked out in algorithmic detail.

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0_3

3.1 Least Squares Problem Formulation

Given some set of experimental data

$$(t_i, y_i), \quad t_i \in \mathbb{R}, y_i \in \mathbb{R}^d, \quad i = 1, \ldots, M \ ,$$

assume that these data can be described by some underlying *law* such that

$$y(t) = \varphi(t; p), \quad y_i = y(t_i).$$

Herein φ is a given *model function* that contains q unknown *(model) parameters*

$$p = (p_1, \ldots, p_q) \ .$$

Usually, we have $dM \gg q$, so that replacing the data by the model would result in some *data compression*, once the parameters are known. At first glance, one might want to determine p in such a way that

$$y_i = \varphi(t_i; p), \quad i = 1, \ldots, M,$$

or, in other words, that the *residuals* vanish, i.e.

$$f_i := y_i - \varphi(t_i, p) = 0, \quad i = 1, \ldots, M. \tag{3.1}$$

In general, however, this will be asking for too much, since both measurement errors and model errors will play a role. Hence, the above relation (3.1) will need to be replaced by some

$$f_i = y_i - \varphi(t_i, p) \approx 0, \quad i = 1, \ldots, M \ .$$

Obviously, the question arises: What does "approximately zero" mean?

Gaussian Least Squares Problem

In this situation, the German mathematician C.-F. Gauss (1777–1855) suggested to minimize the sum of *squares*

$$F^T F = \|F\|_2^2 = \sum_{i=1}^{M} \|f_i\|^2 = \min \quad \text{where} \quad F = (f_i) = (y_i - \varphi(t_i, p)), \ f_i \in \mathbb{R}^d \ .$$

For an illustration of such a minimization see Fig. 3.1.

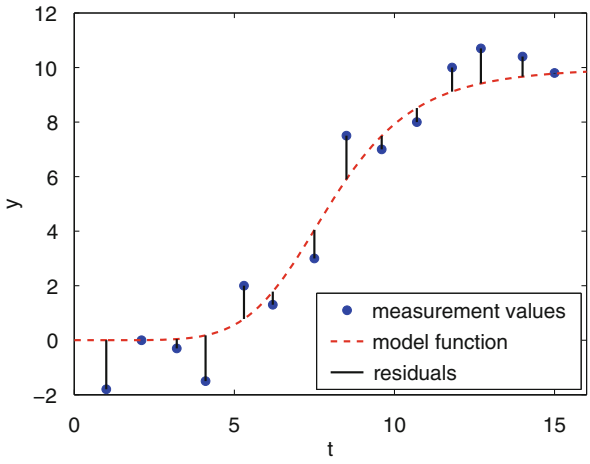


Fig. 3.1 Model compared with data in Gaussian least squares problem ($d = 1$)

Linear Versus Nonlinear Least Squares Problems

Whenever the model function φ depends linearly on the parameters p (not on t!), then we speak of *linear* least squares problems, which we will treat in Sect. 3.2. Whenever this dependence is nonlinear, we speak of *nonlinear* least squares problems, which we will treat in the subsequent Sect. 3.3. In systems biology, the model function $\varphi(t; p)$ is typically defined via a system of ODEs, a numerically more challenging case, which we will work out to necessary detail in Sect. 3.4 further below. The staircase of the presentations has its reason in the following structure: The ODE case leads to nonlinear least squares problems; important algorithmic features for the treatment of nonlinear least squares problems already come up in the treatment of linear least squares problems.

Statistically Correct Formulation

The statistical assumption underlying the Gaussian least squares problem formulation is only valid, if the ansatz is coupled with *measurement tolerances* $0 < \delta y_{ij} < \infty$ such that

$$\|D_{\text{tol}}^{-1} F\|_2^2 = \sum_{i=1}^{M} \sum_{j=1}^{d} \left(\frac{f_{ij}}{\delta y_{ij}} \right)^2 = \min \tag{3.2}$$

with properly defined diagonal matrix D_{tol} in terms of given elements δy_{ij}. If an individual measurement is unavailable, then this can be formally included as $\delta y_{ij} = \infty$, which means that the corresponding term in the sum is just dropped. If measurements could be assumed to be *exact*, then this would formally show up as $\delta y_{ij} = 0$; this does, of course, not mean to divide by zero (!), but that special

algorithms for *equality constrained* linear least squares problems must be applied (interested readers may want to look up [15, Section 4.3]). However, this latter case of exact measurements will practically not occur in systems biology.

The image space of the mapping F has dimension $m \leq M \cdot d$, where equality holds, if measurements are available for all variables at all points. For ease of presentation, we will subsequently quite often use the simpler "one index" version with $f_i \in \mathbb{R}^1$, i.e. $d = 1$.

Typical Measurement Tolerances

Typical choices for the components δy_i are (dropping the second index, as stated above):

(a) *absolute measurement errors*

$$\delta y_i = \delta y_{\text{abs}} ,$$

wherein the common factor δy_{abs} does not need to be included in the problem formulation, since it would just cancel throughout the computation; hence, if no explicit weighting is included, then this is automatically equivalent to having imposed an absolute error concept;

(b) *relative measurement errors* arise whenever the tolerances satisfy

$$\delta y_i = |y_i| \, \text{TOL} ,$$

where the common factor TOL again does not need to be included in the problem formulation, since it would cancel;

(c) *Poisson distribution errors*, e.g., in photodetectors or alike, where all data y_i are positive, give rise to

$$\delta y_i = \sqrt{y_i} .$$

Without an explicit inclusion of such tolerances the whole problem formulation may be wrong in view of the statistical background! Only for ease of writing, we will subsequently mostly drop any kind of weighting.

Inequality Constraints for Parameters

Quite often, parameters are restricted due to inequality constraints, most often as *positivity constraints*. Any such constraints are tractable via nonlinear transformations, say $\phi : u \to p$. In Table 3.1, we give a list of possibilities. In order to illustrate the case, we give a small example below.

Example 10 (Arrhenius law) Assume that $m = 21$ measurements ($d = 1, m = M$)

$$(T_i, K_i) \quad i = 1, \ldots, m$$

Table 3.1 Possible transformations to treat inequality constraints for parameters

Constraint	Transformation $\phi(u)$
$p > 0$	$p = \exp(u)$, see, e.g., formula (3.3)
$A \le p \le B$	$p = A + \frac{B-A}{2}(1 + \sin u)$
$p \le C$	$p = C + \left(1 - \sqrt{1+u^2}\right)$
$p \ge C$	$p = C - \left(1 - \sqrt{1+u^2}\right)$

of reaction rate coefficients K versus temperature T are given. The underlying chemical model is the Arrhenius law (compare (1.15))

$$K(T) = A \cdot \exp\left(-\frac{\Delta E}{RT}\right) , \tag{3.3}$$

where $R = 8.3145 \frac{\mathrm{J}}{\mathrm{mol \cdot K}}$ denotes the universal gas constant, A the unknown pre-exponential factor, and ΔE the unknown activation energy. So the parameter vector is $p = (A, \Delta E)$ and hence $q = 2$. Compared with Fig. 3.1, the temperature T takes the role of time t and K the one of y. As the model function $K(T)$ depends nonlinearly on the parameter ΔE, we have a *nonlinear least squares problem*. Its solution, i.e. model versus data, is depicted in Fig. 3.2, left.

If, however, we take the log of both sides of (3.3), we arrive at the model

$$y(t) := \log K(T) = \log A - \frac{1}{RT}\Delta E . \tag{3.4}$$

Herein the unknown parameters are $p = (\log A, \Delta E)$, again with $q = 2$. Consequently, we transform the data to

$$(T_i, y_i) \quad i = 1, \ldots, m .$$

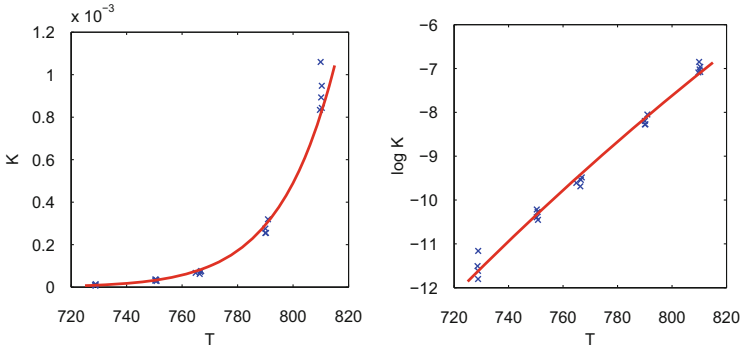


Fig. 3.2 Example 10. *Arrhenius law* ($d = 1, q = 2$) (Data from [17, Exercises 3.7 and 3.8]). *Left:* Nonlinear least squares problem. *Right:* Linear least squares problem

The model function $y(T)$ depends only linearly on both parameters, which means that we have a *linear least squares problem*. The corresponding solution, model versus data, is represented graphically in Fig. 3.2, right.

Non-Gaussian Formulations

Choices other than the Gaussian least squares formulation are:

(a) l_1*-minimization*:

$$\|F\|_1 = \sum_{i=1}^{m} |f_i| = \min ,$$

which is often said to be less sensitive to statistical outliers; this approach is equivalent to a *linear programming* problem, which requires a much larger amount of computing effort;

(b) l_∞*-minimization*:

$$\|F\|_\infty = \max_{i=1,\dots m} |f_i| = \min ,$$

also named *Chebyshev fit*, which leads to a *dual linear programming* problem and again requires much more computational effort.

We will, however, restrict our attention to the Gaussian least squares formulation throughout the book.

Bayesian Approach

All of the above formulations, Gaussian or non-Gaussian, treat parameters p as *fixed constants*; this is sometimes called the frequentist approach. In contrast, the Bayesian approach treats the unknown parameters as *random variables*. In this setting, the *conditional* probabilities of the parameters p given data y read

$$\mathbb{P}(p|y) = \frac{\mathbb{P}(y|p)\mathbb{P}(p)}{\mathbb{P}(y)} . \tag{3.5}$$

This is the famous Bayes' theorem. Herein the conditional probability of the data y given parameters p, the *likelihood* $\mathbb{P}(y|p)$, is given by the formula

$$\mathbb{P}(y|p) = \exp\left(-\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{d} \left(\frac{f_{ij}}{\delta y_{ij}} \right)^2 \right) ,$$

wherein the quantities δy_{ij} play the role of user prescribed standard deviations of the measurements, assumed to be normally distributed. Obviously, maximizing the likelihood is equivalent to minimizing (3.2), the sum of squares in the above exponent. As can be seen in formula (3.5), the denominator value $\mathbb{P}(y)$ is just a normalization constant. In order to determine the so-called *posterior* $\mathbb{P}(p|y)$, information about the so-called *prior* $\mathbb{P}(p)$ will be necessary. This usually requires an a-priori assumption. Throughout this book, we take the frequentist point of view. Readers interested in details about the Bayesian point of view may want to look up the rather recent survey [60] by J. Vanlier et al. or, from a deeper mathematical point of view, the elaborate article [59] by A. M. Stuart. Present algorithmic realizations based on Markov chain Monte Carlo (MCMC) techniques suffer from high computational costs and are therefore not suited for the large ODE networks that we envision in our book.

3.2 Linear Least Squares Problems

In this section, we assume that the model function φ depends linearly on the parameters $p = (p_1, \ldots, p_q)$ so that we may write

$$\varphi(t; p) = a_1(t)p_1 + \ldots + a_q(t)p_q$$

wherein $a_1(t), \ldots, a_q(t) : \mathbb{R} \to \mathbb{R}^d$ are given functions. In short-hand notation of linear algebra we may then express the linear least squares problem as

$$\|y - Ap\|_2 = \min\,, \tag{3.6}$$

where $y = (y_1, \ldots, y_m)$ characterizes the measurements and $A = (a_j(t_i))$ is an (m, q)-matrix with $m \geq q$. In what follows, we will identify the norm $\|\cdot\|$ with the Euclidean norm $\|\cdot\|_2$, unless explicitly stated otherwise.

3.2.1 Normal Equations

As a prerequisite to determine the solution of the above minimization problem (3.6) let us first define the *orthogonal complement* of a subspace $U \subset V$ by

$$U^{\perp} = \{w \in V |\, \langle w, u\rangle = 0 \text{ for all } u \in U\}\,,$$

wherein the inner product $\langle\cdot,\cdot\rangle$ is understood to be the Euclidean inner product. Given some vector $v \in V$, the vector $u \in U \subset V$ that minimizes $\|v - u\|$ is just the orthogonal projection of v onto U. For illustration, view Fig. 3.3, left, wherein for schematic representation we have $V = \mathbb{R}^2$, while U is the linear subspace given by

the straight line. Hence we may write:

$$\|v-u\| = \min_{u' \in U} \|v-u'\| \quad \Longleftrightarrow \quad v-u \in U^{\perp}. \tag{3.7}$$

From this geometric insight we may immediately derive the solution of (3.6). For this purpose, we define the *range* of the (m,q)-matrix A by

$$\mathcal{R}(A) = \{Ax, x \in \mathbb{R}^q\} \subset \mathbb{R}^{m-q+1} .$$

With this definition Fig. 3.3, right, is the direct analogue of Fig. 3.3, left. Hence, by virtue of (3.7), we are immediately led to

$$\|y-Ap\| = \min \quad \Leftrightarrow \quad y-Ap \in \mathcal{R}(A)^{\perp} \tag{3.8}$$

From this insight, we obtain

$$\langle y-Ap, Ap\rangle = \langle A^T(y-Ap), p\rangle = 0 \quad \Leftrightarrow \quad A^T(y-Ap) = 0 .$$

The thus arising q equations

$$A^TAp = A^Ty \tag{3.9}$$

are called the *normal equations*. They are uniquely solvable, if the (q,q)-matrix A^TA is *nonsingular*, which is equivalent to the condition that the (m,q)-matrix A has *full column rank* q (for $m > q$).

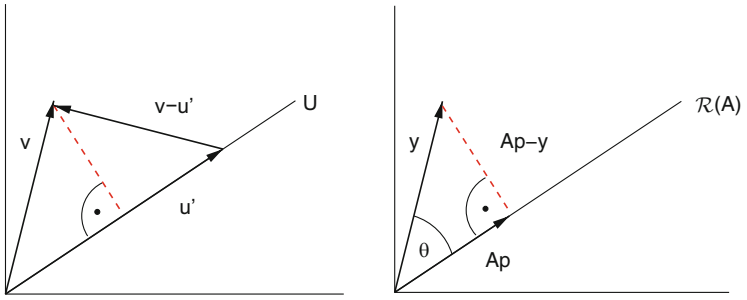


Fig. 3.3 *Left:* General orthogonal projection. *Right:* Geometric solution of linear least squares problem. Note that $\cos\theta = \|Ap\|/\|y\|$

.

Algorithm

In order to solve the above Eqs. (3.9) we first observe that the (q, q)-matrix A^TA is symmetric positive semi-definite. Under the assumption that the (m, q)-matrix A has full column rank q, it is even symmetric positive definite (in short: *spd*). Thus we are naturally led to the following algorithmic steps:

(a) Cholesky factorization $A^TA = LL^T$,
where L is a lower triangular matrix, which is nonsingular, if $\text{rank}(A) = q$,
(b) computation of the right-hand side: $d = A^Ty$,
(c) forward/backward substitution $Lz = d, \quad L^Tp = z$.

The main computational costs of this algorithm are:

- computation of $A^TA : \sim \frac{1}{2}q^2m$,
- Cholesky factorization of $A^TA : \sim \frac{1}{6}q^3$,

which, for $m \gg q$, sum up to total costs of

$$\sim \frac{1}{2}q^2m \text{ operations} . \tag{3.10}$$

However, this algorithm ignores an important aspect, since we know from (A.7) that

$$\text{cond}_2(A^TA) = \text{cond}_2(A)^2 . \tag{3.11}$$

In fact, a typical feature of least squares problems is that $\text{cond}(A) \gg 1$, so that this algorithmic approach drastically deteriorates the condition of the originally stated problem.

Example 11 This illustrative Example is notorious in the literature. Let A denote the following $(3, 2)$-matrix and A^TA the corresponding $(2, 2)$-matrix:

$$A = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}, \quad A^TA = \begin{pmatrix} 1+\varepsilon^2 & 1 \\ 1 & 1+\varepsilon^2 \end{pmatrix}$$

The matrix A has full rank, if $\varepsilon > \text{eps}$, where eps denotes the relative machine precision defined by the floating point operation $fl(1+\varepsilon) = 1$ for $|\varepsilon| \leq \text{eps}$. Suppose, however, we have

$$\text{eps} < \varepsilon \leq \sqrt{\text{eps}} .$$

Then we obtain the rounded matrices

$$fl(A) = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}, \quad fl(A^T A) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$

which means that $fl(A)$ still has full rank, whereas $fl(A^T A)$ is strictly singular. Hence, the normal equations are not solvable! This effect is also clearly reflected in the condition numbers (see Appendix A.2)

$$\mathrm{cond}_2(A^T A) = \frac{2+\varepsilon^2}{\varepsilon^2} \geq \frac{2}{\mathrm{eps}}, \quad \mathrm{cond}_2(A) \doteq \frac{\sqrt{2}}{\varepsilon} < \frac{\sqrt{2}}{\mathrm{eps}} \tag{3.12}$$

3.2.2 QR-Factorization

An alternative algorithmic approach starts from the fact that the Euclidean norm $\|\cdot\|$ is invariant under orthogonal transformation. To see this, let Q denote an orthogonal (m,m)-matrix, i.e. with $QQ^T = I_m$, so that

$$\|y - Ap\|^2 = (y - Ap)^T QQ^T (y - Ap) = \|Q^T (y - Ap)\|^2 .$$

The basic idea is to construct some orthogonal matrix Q such that the rectangular matrix A is transformed to some upper triangular shape

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \Leftrightarrow \quad Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{3.13}$$

where R is an upper triangular (q,q)-matrix. Suppose now that such a factorization has been realized and split the vector according to $Q^T y = (\eta_1, \eta_2)^T$. Then the linear least squares problem can be solved via

$$\begin{aligned} \|y - Ap\|^2 = \|Q^T (y - Ap)\|^2 &= \left\| \begin{pmatrix} \eta_1 - Rp \\ \eta_2 \end{pmatrix} \right\|^2 \\ &= \|\eta_1 - Rp\|^2 + \|\eta_2\|^2 \geq \|\eta_2\|^2 . \end{aligned}$$

Obviously, the unknown parameter p should be chosen such that the first term vanishes. Under the assumption that $\mathrm{rank}(A) = q$, which implies $\mathrm{rank}(R) = q$, the triangular matrix is invertible and the solution is uniquely determined by

$$Rp = \eta_1 \quad \Leftrightarrow \quad p = R^{-1} \eta_1 . \tag{3.14}$$

Thus we arrive at the minimal residual

$$\|F\|_2 = \|y - Ap\|_2 = \|\eta_2\|_2 \,.$$

In passing we note that, unlike the normal equation approach, the QR-factorization approach is numerically stable, since $\mathrm{cond}_2(Q) = 1$ implies $\mathrm{cond}_2(A) = \mathrm{cond}_2(R)$.

As a standard, the QR-factorization is realized via Householder reflections with column pivoting, details of which are elaborated in Appendix A.3.

Example 12 Let us return to the matrix A defined in Example 11 above. In the QR approach we obtain the rounded intermediate results

$$\tilde{Q}^T = \frac{1}{\sqrt{2}} \begin{pmatrix} -\sqrt{2} & -\sqrt{2}\varepsilon & 0 \\ \varepsilon & -1 & 1 \\ -\varepsilon & 1 & 1 \end{pmatrix}, \quad \tilde{R} = \begin{pmatrix} -1 & -1 \\ 0 & \sqrt{2}\varepsilon \\ 0 & 0 \end{pmatrix}$$

For $\varepsilon >$ eps, the triangular linear system (3.14) is *numerically* uniquely solvable, compare (3.12).

Rank Decision

From (3.9) we know that the linear least squares problem is only uniquely solvable, if the (m, q)-matrix A has full column rank q. Unlike the normal equation algorithm, the QR-factorization supplies a cheap possibility to "identify" the rank. In this approach, the rank of A will be the same as the one for the arising matrix R. In general, the question of how to "determine" the rank of a matrix is all but trivial. Strictly speaking, *singular value* decomposition should be exploited: Let

$$\sigma_1 \geq \ldots \sigma_q \geq 0$$

denote the singular values of the matrix A, ordered according to their size. If $\sigma_q > 0$, then (A.6) tells that the condition number can be expressed as

$$\mathrm{cond}_2(A) = \frac{\sigma_1}{\sigma_q} \,.$$

If $\sigma_q = 0$, then the rank of A is formally defined by

$$\sigma_\rho > 0,\ \sigma_{\rho+1} = 0 \quad \Leftrightarrow \quad \rho = \mathrm{rank}(A)\ .$$

In actual computation, however, the *relative accuracy* ε_A of the matrix A will have to come in, see Appendix A.3. A system is then said to be *computationally solvable*, whenever

$$\mathrm{cond}_2(A) = \frac{\sigma_1}{\sigma_q} < \frac{1}{\varepsilon_A} \quad \Leftrightarrow \quad \sigma_q > \sigma_1\ \varepsilon_A\ . \tag{3.15}$$

In this spirit, some *numerical rank* will be defined via

$$\sigma_\rho > \sigma_1\ \varepsilon_A,\ \ \sigma_{\rho+1} \le \sigma_1\ \varepsilon_A \quad \Leftrightarrow \quad \rho = \mathrm{rank}_{\mathrm{SVD}}(A)\ . \tag{3.16}$$

In order to realize this definition numerically, we would need to implement singular value decomposition for the whole linear least squares solution. This is usually regarded as "overkill".

As shown in Appendix A.3, the QR-factorization by Householder transformations with column pivoting generates an upper triangular matrix $R = (r_{ij})$ whose diagonal elements are ordered according to

$$|r_{11}| \ge \ldots \ge |r_{qq}| \ge 0\ . \tag{3.17}$$

If $|r_{qq}| > 0$, we may define some *subcondition number*

$$\mathrm{sc}(A) = \frac{|r_{11}|}{|r_{qq}|}\ ,$$

The name originates from the fact that

$$\mathrm{sc}(A) \le \mathrm{cond}_2(A).$$

This quantity naturally arose from a subtle elementwise error propagation analysis and was defined by P. Deuflhard and W. Sautter [20]. For the subcondition number similar properties as for the condition number hold:

(a) $\mathrm{sc}(A) \ge 1$
(b) $\mathrm{sc}(\alpha A) = \mathrm{sc}(A), \quad \alpha \in \mathbb{R}^1, \alpha \ne 0$
(c) $\mathrm{sc}(A) = \infty \quad \Leftrightarrow \quad A \ne 0$ singular

Example 13 For illustration, we return to Example 11 and compare

$$\mathrm{cond}_2(R) \doteq \sqrt{2}/\varepsilon < \sqrt{2}/\varepsilon_A\ , \quad \mathrm{sc}(R) = 1/(\sqrt{2}\ \varepsilon) < 1/(\sqrt{2}\ \varepsilon_A)\ ,$$

which shows that the (much cheaper computable) subcondition number differs only by a factor of 2 from the condition number.

Based on the subcondition number, a system is said to be *computationally solvable*, whenever (see [15] and references therein)

$$\mathrm{sc}(A) = \frac{|r_{11}|}{|r_{qq}|} < \frac{1}{\varepsilon_A} \quad \Leftrightarrow \quad |r_{qq}| > |r_{11}|\varepsilon_A \ . \tag{3.18}$$

Computational experience in practical applications shows that the difference between the two criteria (3.15) and (3.18) is marginal. In addition, the inequalities

$$\frac{1}{\varepsilon_A} < \mathrm{sc}(A) \leq \mathrm{cond}_2(A)$$

imply that "numerically singular" due to QR-factorization is always "numerically singular" due to SVD, but not vice versa.

The extension to define some reduced *numerical rank* will naturally be

$$|r_{\rho\rho}| > |r_{11}|\varepsilon_A,\ |r_{\rho+1,\rho+1}| \leq |r_{11}|\varepsilon_A \quad \Leftrightarrow \quad \rho = \mathrm{rank}_{\mathrm{QR}}(A) \ . \tag{3.19}$$

Note that the rank decision devices (3.16) and (3.19) are not as closely connected as (3.15) and (3.18).

Generally speaking, an "exact" rank ρ cannot be determined, among other reasons due to the not precisely known accuracy ε_A. If the matrix comes from pure linear algebra, then mostly the machine precision eps comes in via $\varepsilon_A = \mathrm{eps}$. In Sect. 3.4 below, the matrix will arise as a Jacobian that is elementwise computed via the solution of a system of ODEs, the sensitivity equations; as a consequence, the accuracy ε_A will there be the discretization error for these ODEs, always much larger than the machine precision.

Influence of Row and Column Scaling

The criteria for both computational solvability and rank decision are based on the values of the diagonal elements of the upper triangular matrix R. As a consequence of the QR-factorization worked out in Appendix A.3, these values depend *nonlinearly* on row as well as column scaling. As for *row scaling*, recall from (3.2) that

$$D_{\mathrm{tol}} = \mathrm{diag}(\delta y_1, \ldots, \delta y_m) > 0$$

needs to be chosen to ensure that the problem at hand is correctly formulated in terms of statistics. As for *column scaling*, this is induced by scaling different

physical units of the parameters, say

$$D_{\text{scal}} = \text{diag}(p_{1,\text{scal}}, \ldots, p_{q,\text{scal}}) \; > \; 0 \; .$$

Hence, in actual computation, the linear least squares problem (3.6) should always be envisioned in its scaled form as

$$\|D_{\text{tol}}^{-1}(AD_{\text{scal}}D_{\text{scal}}^{-1}p - y)\| = \|\bar{A}\bar{p} - \bar{y}\| = \min \tag{3.20}$$

where

$$\bar{A} = D_{\text{tol}}^{-1}AD_{\text{scal}}, \quad \bar{y} = D_{\text{tol}}^{-1}y, \quad \bar{p} = D_{\text{scal}}^{-1}p \; .$$

In this setting, the QR-factorization will be performed on the weighted input $\bar{A}\Pi$ (including the rank decision, see above) to obtain the scaled parameters $\bar{p}$, from which finally the parameters

$$p = \Pi^T D_{\text{scal}} \, \bar{p}$$

are obtained. For the variation of the scaled variable components we get

$$\delta\bar{y}_i = \delta\frac{y_i}{\delta y_i} = \frac{\delta y_i}{\delta y_i} = 1, \quad i = 1, \ldots, m,$$

i.e. all scaled variables have the same *statistical standard deviation* with respect to some reasonably defined underlying statistical distribution. This justifies to use the condition number $\text{cond}(\bar{A})$ (or the subcondition number $\text{sc}(\bar{A})$, respectively) as a measure of errors only of the matrix $\bar{A}$, ignoring those in the right hand-side $\bar{y}$.

Algorithm for Rank-Deficient Case

Suppose that, based on the decision (3.19), the remainder "small" part of the decomposed matrix has been dropped, see (A.9), so that we have

$$Q^T A\Pi = \begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix} \; , \quad Q^T y = (\eta_1, \eta_2)^T \; . \tag{3.21}$$

We are therefore left to solve the *underdetermined* system

$$\|Ap - y\|_2^2 = \|[R, S] \; \Pi^T p - \eta_1\|_2^2 + \|\eta_2\|_2^2 = \min \; .$$

Among the various efficient possibilities, we here select the $QR\hat{Q}$-factorization due to G. Peters and J.H. Wilkinson [50] because of its simple interpretation. Its basic

idea is to construct Householder transformations $\hat{Q}$ applied from the right in such a way that

$$[R, S]\ \hat{Q}^T \hat{Q}\ \Pi^T p = \eta_1 \quad \Leftrightarrow \quad [\hat{R}, 0]\ \hat{Q}\ \Pi^T p = \eta_1\ .$$

Suppose we decompose $\hat{p} := \hat{Q}\Pi^T p = (\hat{p}_1, \hat{p}_2)^T$ so that we have $\|\hat{p}\| = \|p\|$. With this factorization, the minimization problem now reads

$$\|\hat{R}\ \hat{p}_1 - \eta_1\|^2 + \|\hat{p}_2\|^2 = \min\ .$$

The upper triangular matrix $\hat{R}$ differs from R only in its diagonal elements $\hat{r}_{kk}$. Consequently, an order comparable to (A.11) no longer holds, so that a subcondition number based on the diagonal elements $\hat{r}_{kk}$ would no longer be a valid concept. These diagonal elements satisfy the inequality

$$\hat{r}_{kk} \geq r_{kk}\ , \quad k = 1, \ldots, \rho$$

which guarantees that $\hat{R}$ is nonsingular. Hence, the *shortest* linear least squares solution is uniquely defined as

$$\hat{p}_1 = \hat{R}^{-1}\eta_1, \quad \hat{p}_2 = 0\ .$$

The storage scheme is similar to Fig. A.1, only the diagonal elements $r_{kk} = \alpha_k$ are replaced by $\hat{r}_{kk} = \hat{\alpha}_k$.

Model Reduction

The above algorithm conveniently permits to shed some light into linear dependencies of the original parameters p. If we go back to the original meaning of the matrix $A = \dfrac{\partial \varphi}{\partial p}$, we see that the above factorization implies

$$\frac{\partial \varphi(t_i; p)}{\partial \hat{p}_2} \approx 0\ , \quad i = 1, \ldots, M\ . \tag{3.22}$$

In words: In the light of the given measurements, the model function φ does not depend on the parameters $\hat{p}_2$, which are linear combinations of the original parameters p. From this interpretation, we may derive some model reduction technique. For this purpose, let us define

$$\hat{I}_\rho = \mathrm{diag}(1, \ldots, 1, 0, \ldots 0), \quad \hat{I}_\rho^\perp = I_q - \hat{I}_\rho = \mathrm{diag}(0, \ldots, 0, 1, \ldots 1)\ .$$

Hence,

$$\hat{I}_\rho \hat{p} = \hat{p}_1, \quad \hat{I}_\rho^\perp \hat{p} = \hat{p}_2 \, . \tag{3.23}$$

Assume now that the original parameters were dependent as

$$\hat{I}_\rho^\perp \hat{Q} \Pi^T p = 0 \, , \tag{3.24}$$

then the shortest solution condition $\hat{p}_2 = 0$ would be automatically satisfied. This would suggest some reduction of the number of parameters from q to $\rho < q$.

3.2.3 Generalized Inverses

Suppose we have to solve a *linear system* $Ap = y$ for a (q, q)-matrix A. Under the assumption that A is nonsingular, we are used to write the solution as $p = A^{-1}y$ – of course, understood to be a formal notation only, not meant to be naively realized in actual computation! The inverse is defined by the formal properties

$$A^{-1}A = I, \quad AA^{-1} = I \, . \tag{3.25}$$

Suppose now that we have to solve a *linear least squares problem* for some given (m, q)-matrix A

$$\|Ap - y\| = \min, \quad \rho = \operatorname{rank}(A) \leq q < m \, .$$

In this case we would again like to have a formal definition of some *generalized inverse* A^+ such that the solution can be formally written in a unique manner as

$$p = A^+ y \, .$$

Let us study the situation in terms of the rank ρ of the matrix A.

Full Rank

Assume first that $\rho = q$. Then A^TA is nonsingular and the unique least squares solution can be written as

$$A^TAp = A^Ty \quad \Leftrightarrow \quad p = \underbrace{(A^TA)^{-1}A^T}_{=A^+} y \, ,$$

where a definition of the generalized inverse naturally pops up. Again, the formal representation $p = A^+ y$ is understood to be independent of the algorithm actually used to solve the problem, which in our case would better be the *QR*-approach than the normal equations approach.

Rank Deficiency

Next, let $\rho < q$. In this case, a "unique" solution no longer exists. To see this, let us introduce the *nullspace* $\mathcal{N}(A)$ of the matrix A by

$$\mathcal{N}(A) = \{z \in \mathbb{R}^q : Az = 0\}.$$

Then for any solution p further solutions $p + z, z \in \mathcal{N}(A)$ exist, since

$$\|y - Ap\| = \|y - A(p + z)\|, \quad z \in \mathcal{N}(A)$$

In other words: The solutions form a $(q - \rho)$-dimensional subspace

$$\mathcal{L}(y) = \{p \in \mathbb{R}^q : \|y - Ap\| = \min\} = p^* + \mathcal{N}(A) \ .$$

Unlike the solutions, the *residual* is unique, which means that it is the same for all solutions $p \in \mathcal{L}(y)$, since

$$r(p) = y - Ap = y - A(p^* + z) = y - Ap^* - Az = y - Ap^* = r(p^*) \ . \tag{3.26}$$

This means that all different solutions have exactly the same "data fit" and cannot be distinguished by mere comparison with the given data! Though this insight is crucial for any modelling, it is nevertheless quite often ignored by researchers.

From the subspace $\mathcal{L}(y)$, one usually selects the "shortest" solution

$$p^* = A^+ y$$

by the condition

$$\|p^*\| \leq \|p\| \quad \text{for all} \quad p \in \mathcal{L}(y) \ .$$

The geometrical situation is represented in Fig. 3.4.

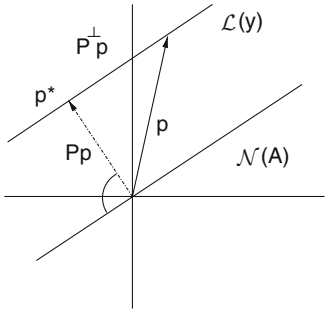


Fig. 3.4 "Shortest" solution p^* out of the solution set $\mathcal{L}(y) = p^* + \mathcal{N}(A)$ in terms of the nullspace projection operator $P^{\perp} : \mathbb{R}^d \to \mathcal{N}(A)$ as defined in the text

Moore-Penrose Axioms

In the full rank as well as in the rank-deficient case the four axioms due to [49] hold:

(i) $(A^+A)^T = A^+A$
(ii) $(AA^+)^T = AA^+$
(iii) $A^+AA^+ = A^+$
(iv) $AA^+A = A$

These four axioms uniquely define the generalized inverse A^+ – for a proof see the original paper [49] or, e.g., the textbook [17]. Let us further define the matrices

$$P := A^+A, \quad \bar{P} := AA^+ . \tag{3.27}$$

From axioms (i) and (ii) we see that

$$P^T = P, \quad \bar{P}^T = \bar{P} .$$

From axioms (iii) and (iv) we get

$$P^2 = (A^+A)(A^+A) = A^+A = P, \quad \bar{P}^2 = (AA^+)(AA^+) = AA^+ = \bar{P} .$$

This means that the two matrices P and $\bar{P}$ are *projection matrices*. Their orthogonal complements

$$P^{\perp} = I - P, \qquad \bar{P}^{\perp} = I - \bar{P}$$

satisfy

$$A^+\bar{P}^{\perp} = A^+(I - AA^+) = 0, \quad P^{\perp}A^+ = (I - A^+A)A^+ = 0 . \tag{3.28}$$

The matrix $\bar{P}$ performs a splitting of the residual space, since, in the here introduced notation,

$$r(p^*) = y - Ap^* = y - AA^+y = (I - \bar{P})y = \bar{P}^\perp y \ .$$

The matrix P performs a splitting of the solution space $\mathcal{L}(y)$, since for any $p \in \mathcal{L}(y)$

$$p^* = A^+(Ap + r) = (AA^+)p + A^+r = Pp + A^+\bar{P}^\perp y = Pp \ .$$

From this it is easy to verify that p^* is shorter than any other solution, since

$$\|p^*\|^2 = \|Pp\|^2 \le \|Pp\|^2 + \|P^\perp p\|^2 = \|p\|^2 \ , \quad p \in \mathcal{L}(y) \ .$$

Of course, upon including column scaling D_{scal} into the consideration, this property will be replaced by

$$\|D_{\text{scal}}^{-1}p^*\| \le \|D_{\text{scal}}^{-1}p\| \ .$$

Note that here column pivoting does not play a role, since the norm $\|\cdot\|$ is invariant under permutation.

Remark 10 In the notation of the preceding Sect. 3.2.2, some straightforward calculation yields the connection with the $QR\hat{Q}$-factorization in the form

$$\bar{P} = AA^+ = Q\hat{I}_\rho Q^T, \quad P = A^+A = \Pi\hat{Q}^T\hat{I}_\rho\hat{Q}\Pi^T \ . \tag{3.29}$$

Obviously, both projectors have rank ρ. As a consequence, the relation (3.24) for model reduction can be written in the form

$$P^\perp p = \Pi\hat{Q}^T(0, \hat{p}_2)^T = 0 \ . \tag{3.30}$$

Linear Equality Constraints

The QR-factorization as worked out in Appendix A.3 can be easily modified in such a way that also linear equality constraints with $\delta y_i = 0$ can be tackled. In this limiting case, there still exist projections $P, P^\perp$ in the domain space of the matrix A, but no longer any projections $\bar{P}$ in the image space of A. Interested readers may find details in [15, Section 4.1.2].

3.3 Nonlinear Least Squares Problems

We return to the original problem (3.2), but drop all double indices and all (necessary!) measurement tolerances for ease of presentation. Here we merely study the *nonlinear* least squares problem

$$F(p)^T F(p) = \|F(p)\|^2 = \min \tag{3.31}$$

in terms of some nonlinear mapping $F : D \subseteq \mathbb{R}^q \to \mathbb{R}^m$ with $q < m \leq M \cdot d$, explicitly written as

$$F(p) = [F_1(p), \ldots, F_M(p)]^T \quad \text{with} \quad F_i(p) = y_i - \varphi(t_i, p) \ ,$$

if all components have been measured, i.e. $m = dM$; otherwise only part of the above components are inserted. The problem is to find a solution p^* such that

$$\|F(p^*)\|^2 = \min_p \|F(p)\|^2.$$

Whenever

$$F(p^*) = 0, \tag{3.32}$$

then the problem is said to be *compatible*. As in the linear case, this occurrence is rather rare, while "nearly compatible" problems are quite usual – a term to be made more precise below.

3.3.1 Local Newton Versus Gauss-Newton Approach

We next turn to the numerical solution of such problems. To start with, we define the corresponding Jacobian (m, q)-matrix, as

$$F'(p) = \begin{pmatrix} \frac{\partial}{\partial p_1} F_1(p) & \cdots & \frac{\partial}{\partial p_q} F_1(p) \\ \vdots & & \vdots \\ \frac{\partial}{\partial p_1} F_m(p) & \cdots & \frac{\partial}{\partial p_q} F_m(p) \end{pmatrix} = - \begin{pmatrix} \frac{\partial}{\partial p_1} \varphi(t_1; p) & \cdots & \frac{\partial}{\partial p_q} \varphi(t_1; p) \\ \vdots & & \vdots \\ \frac{\partial}{\partial p_1} \varphi(t_M; p) & \cdots & \frac{\partial}{\partial p_q} \varphi(t_M; p) \end{pmatrix}$$

If we write $g(p) = \frac{1}{2} F(p)^T F(p) = \min$, then $g'(p)$ can be calculated componentwise as

$$g'(p) = F'(p)^T F(p)$$

As a sufficient condition for p^* to be a local minimum we thus get

$$g'(p^*) = F'(p^*)^T F(p^*) = 0 \quad \text{and} \quad g''(p^*) \text{ positive definite}$$

So we might approach the problem solution by applying the Newton method to solve the following system of q nonlinear equations

$$G(p) := F'(p)^T F(p) = 0\ . \tag{3.33}$$

Local Newton Method

Rather than working on the nonlinear least squares problem (3.31), this method tries to solve the gradient system of nonlinear equations (3.33) by successive linearization, i.e. by an iterative *sequence of linear equations*. This approach requires the Jacobian (q, q)-matrix

$$G'(p) = F'(p)^T F'(p) + F''(p)^T F(p)\ ,$$

wherein the term $F''(p)$ is a tensor so that the product $F''(p)^T F(p)$ represents some larger amount of computational work.

The basic idea is to approximate G by the linear part of its Taylor series expansion in a neighborhood around a starting guess p^0, i.e.

$$G(p) = G(p^0) + G'(p^0)(p - p^0) + \mathcal{O}(\|p - p^0\|^2)$$

Then the first iterate can be computed according to

$$G'(p^0)\Delta p^0 = -G(p^0), \quad p^1 = p^0 + \Delta p^0$$

Upon repeating this procedure (assuming the Jacobians $G'(p)$ are nonsingular fora ll arising arguments p), we arrive at the *Newton iteration*

$$\Delta p^k = -G'(p^k)^{-1} G(p^k), \quad p^{k+1} = p^k + \Delta p^k, \quad k = 0, 1, \ldots \tag{3.34}$$

This iteration is known to converge *quadratically* in a sufficiently small neighborhood of a unique solution point, which is why (3.34) is named *local Newton method*.

Local Gauss-Newton Method

In this approach, we start from the observation that in the compatible case, i.e. when $F(p^*) = 0$, one has $G'(p^*) = F'(p^*)^T F'(p^*)$ so that the undesirable evaluation of the tensor term $F''(p)^T F(p)$ might be omitted in some neighborhood of p^*. The same argument might also hold in the "almost compatible" case, i.e. when $F(p^*)$

is "small" (in some sense specified in Appendix A.4). With this simplification, we arrive at the iteration

$$F'(p^k)^T F'(p^k)\Delta p^k = -F'(p^k)^T F(p^k), \quad p^{k+1} = p^k + \Delta p^k, \quad k = 0, 1, \ldots .$$

By comparison with the preceding Sect. 3.2, we detect that this corresponds to solving an iterative *sequence of linear least squares problems* of the kind

$$\|F'(p^k)\Delta p^k + F(p^k)\| = \min, \quad k = 0, 1, \ldots .$$

This is why this iteration is usually called the *Gauss-Newton iteration*.[1] In view of the above Sect. 3.2.3 we may write it as

$$\Delta p^k = -F'(p^k)^+ F(p^k), \quad p^{k+1} = p^k + \Delta p^k, \quad k = 0, 1, \ldots \tag{3.35}$$

in terms of some generalized inverse. Thus we automatically include the treatment of the rank-deficient case as well. From (A.26) in Appendix A.4 we take that this method converges *locally* in the sense that

$$\|\Delta p^k\| \leq \frac{\omega}{2}\|\Delta p^{k-1}\|^2 + \kappa(p^{k-1})\|\Delta p^{k-1}\| . \tag{3.36}$$

Herein the first term represents the *quadratic convergence* part as in Newton's method, while the second term with the *incompatibility factor* κ represents some *asymptotic linear convergence* of the Gauss-Newton method. Obviously, a necessary condition for local convergence is that

$$\kappa(p) < 1 \quad \text{in some neighborhood of a solution point } p^* , \tag{3.37}$$

see Appendix A.4 for details of the interpretation. This property defines the term *adequate* nonlinear least squares problems. In summary:

The Gauss-Newton method converges for adequate nonlinear least squares problems in a sufficiently small neighborhood of a solution point p^*.

That is why (3.35) is called *local Gauss-Newton method*.

Termination Criterion

On the basis of the statistical interpretation of the Gauss-Newton method, it is suggested to terminate the iteration as soon as linear convergence dominates the iteration process: iteration beyond that point would just lead to an accuracy far below reasonable in comparison between model and data. Following [15] and in

[1] In the statistics community the Gauss–Newton method is often named *scoring method*.

view of (3.47), the criterion

$$\|\overline{\Delta p}^{k+1}\| \leq \text{PTOL} \tag{3.38}$$

is applied in terms of some user prescribed error tolerance PTOL. After termination in the linear convergence phase, the *finally achieved accuracy* will then be

$$\|p^{k+1} - p^*\| \approx \frac{\kappa(p^k)}{1 - \kappa(p^k)} \|\Delta p^k\| \,. \tag{3.39}$$

This criterion prevents codes from turning inefficient, if unaware users require a too stringent error tolerance threshold PTOL. (Of course, the norms are understood to be *scaled* norms.)

Model Reduction

Suppose we are in the rank-deficient setting. Let us partition the transformed parameters $\hat{p}$ according to (3.23) so that

$$\hat{I}_\rho \Delta\hat{p} = \Delta\hat{p}_1, \quad \hat{I}_\rho^\perp \Delta\hat{p} = \Delta\hat{p}_2 \,. \tag{3.40}$$

Upon transferring (3.22) to the nonlinear case where now A is the Jacobian matrix $F'(p) = -\dfrac{\partial\varphi}{\partial p}$, we again see that

$$\frac{\partial\varphi(t_i; p)}{\partial\hat{p}_2} \approx 0 \,, \quad i = 1, \ldots, M \,. \tag{3.41}$$

Once more, this indicates that the model function φ does not depend on the parameters $\hat{p}_2$ or, alternatively, that the already obtained value of $\hat{p}_2$ is already determined to sufficient accuracy so that the decision $\Delta\hat{p}_2 = 0$ makes sense. Upon transferring the result (3.24) from linear least squares problems to the nonlinear case we obtain

$$\hat{I}_\rho^\perp \hat{Q}_k \Pi_k^T \Delta p^k = 0 \,, \tag{3.42}$$

which indicates a *locally* linear dependence of the original parameters, which could also indicate some globally nonlinear dependence.

"Simplified" Gauss-Newton Method

Now and then, computational scientists are tempted to "save computing cost" by keeping the initial pseudo-inverse, i.e. setting $F'(p)^+ = F'(p^0)^+$ throughout the

algorithm, just as in the simplified Newton method. However, unlike the simplified Newton method, such a simplified Gauss-Newton method would actually solve the *wrong* nonlinear least squares problem

$$\|F'(p^0)F'(p^0)^+F(p)\| = \min \Leftrightarrow F'(p^0)^+F(\bar{p}) = 0 \quad \text{with solution} \quad \bar{p} \neq p^*.$$

Optimization Methods

The same undesirable effect as above comes up, if certain "derivative-free" optimization routines are applied. In this case, too, the wrong problem is often solved. In fact, one should always keep in mind that *nonlinear least squares problems are very special optimization problems with a statistical background* to be carefully observed!

3.3.2 Globalization of Gauss-Newton Method

In order to get rid of the restriction that the initial guess p^0 must be "sufficiently close" to the solution point p^*, globalization techniques are usually applied, which are able to tackle a larger set of problems.

Levenberg-Marquardt Method

This is probably the most popular globalization method.[2] Just like Newton's method, this approach starts from some linearization around the current iterate, say p^k, with, for the time being, arbitrary correction Δz:

$$F(p^k + \Delta z^k) \doteq F(p^k) + F'(p^k)\Delta z^k,$$

The basic idea now is that linearization only holds in a "small" neighborhood. Therefore it is restricted to some neighborhood of p^k defined by

$$\|\Delta z^k\| \leq \delta_k \ ,$$

where the iterative *trust region parameters* δ_k are chosen appropriately. By application of iterative *Lagrange multipliers* $\gamma_k > 0$, we get the modified minimization problem

$$\|F(p^k) + F'(p^k)\Delta z^k\|^2 + \gamma_k \|\Delta z^k\|^2 = \min \ .$$

[2] Also called Tikhonov-Phillips regularization in the general context of inverse problems.

By taking formal derivatives, we arrive at the Levenberg-Marquardt iteration

$$(F'(p^k)^T F'(p^k) + \gamma_k I_q)\Delta z^k = -F'(p^k)^T F(p^k) \ , \quad k = 0, 1, \ldots \ .$$

As there are connections between the parameters γ_k and δ_k, the choice of one of them is sufficient. Properties of this iteration are:

(a) For $\gamma_k \to 0$, the Levenberg-Marquardt iteration merges into the local Gauss-Newton iteration, which may, also for the rank-deficient case, formally be written as

$$F'(p)^+ = \lim_{\gamma \to 0} (F'(p)^T F'(p) + \gamma I)^{-1} F'(p)^T \ . \tag{3.43}$$

Recall that the Gauss-Newton method converges for "adequate" nonlinear least squares problems only, as has been described in Appendix A.4. Hence, if the solution is locally unique and adequate, the Levenberg-Marquardt method will work, too, if only the parameters approach 0 when the iterates approach p^*.

(b) For $\gamma_k > 0$, the matrix $(F'(p^k)^T F'(p^k) + \gamma_k I_q)$ is always regular, which is the reason for the popularity of the method. This case arises, too, if there is not enough data available to identify the model parameters. In other words, the method is "masking" the possible occurrence of a non-uniqueness of solutions. This information, however, is of utmost importance in any scientific modelling: If the computed "solution" for a parameter is not unique, it cannot be trusted in situations other than the one just used to identify it.

(c) The Levenberg-Marquardt iteration has a tendency to terminate numerically at some point $\bar{p}$, where the gradient of the functional $g(p)$ is "small", i.e.

$$g'(\bar{p}) = F'(\bar{p})^T F(\bar{p}) \approx 0 \ . \tag{3.44}$$

Note, however, that the final iterate $\bar{p}$ might be just a "stationary" point rather than a solution point.

Summarizing, the method always offers a "numerical solution", which may be the reason, why it is so popular. However, this might even occur, when there is no close-by solution point at all! Thus the method is prone to lead to misinterpretations in terms of the underlying scientific model.

Global Gauss-Newton Method

Basic theoretical features of this method are worked out in Appendix A.4. Here we want to essentially deal with the details necessary to be known by scientists using the corresponding algorithms. From (A.29), the global Gauss-Newton iteration reads

$$p^{k+1} - p^k = \lambda_k \Delta p^k, \quad \Delta p^k = -F'(p^k)^+ F(p^k), \quad k = 0, 1, \ldots \tag{3.45}$$

The factors $0 < \lambda_k \leq 1$ are chosen according to some adaptive *damping strategy* (for an explanation see [15]) such that the *monotonicity criterion*

$$\|\overline{\Delta p^{k+1}}\| \leq \|\Delta p^k\|, \quad \text{where} \quad \overline{\Delta p^{k+1}} = -F'(p^k)^+ F(p^k + \lambda_k \Delta p^k) \tag{3.46}$$

is met; herein Δp^k are the *ordinary* Gauss-Newton corrections, $\overline{\Delta p^{k+1}}$ the *simplified* Gauss-Newton corrections, which are only computed for the purpose of comparison within the above test. In order to improve convergence, some *rank strategy* may also be applied, for details see [15]. In passing, we take from (A.31) that

$$\|\overline{\Delta p^{k+1}}\| \leq \frac{\omega}{2} \|\Delta p^k\|^2 , \tag{3.47}$$

i.e. the simplified Gauss-Newton corrections mimic the quadratic asymptotic convergence behavior of Newton's method.

Iteration Exits

There is a variety of exits from the Gauss-Newton method, which reflect the intricate complexity of nonlinear least squares problems. This complexity should by no means be underestimated, because otherwise the whole scientific modelling may be of minor value.

In the *full rank* case $\rho = q$, the following two exits may occur:

(a) For *adequate* problems, the damping factors λ_k approach the value 1 as soon as the iterates are "sufficiently close" to a solution point p^*. In this case, the final accuracy of the solution may be detailed by an a-posteriori perturbation analysis described below.
(b) For *inadequate* problems, the damping factors approach a value $\lambda_k \approx 1/\kappa$ without convergence. In this case, the model should be improved; for details see [15] and references therein.

In the *rank-deficient* case $\rho < q$, there is no "unique" solution, even though the iteration may converge. Then, apart from the numerical output solution p^*, there exists some local solution subset that can be characterized by

$$p \in \{p = p^* + P^{\perp} z \quad \text{for arbitrary} \quad z \in \mathbb{R}^q\} . \tag{3.48}$$

To see this, just linearize the mapping $F(p)$ around p^*, i.e.

$$F(p) = F(p^*) + F'(p^*)(p - p^*) + \mathcal{O}(\|p - p^*\|^2) .$$

Now, restrict p according to (3.48) and use $F'(p^*)P^{\perp} = 0$ from (3.28), which then implies

$$F(p) = F(p^*) + \mathcal{O}(\|p - p^*\|^2) \,. \tag{3.49}$$

This relation is the nonlinear extension of property (3.26) for linear least squares problems. In other words: Even though the solution may turn out to be non-unique, the observable nonlinear residuals $F(p)$ are essentially unique for all solution points, i.e. they supply the same kind of possibly "nice" plots when comparing data and model.

Example 14 (Verhulst population model [61]) This example deals with fitting a model to human population data, see [45]. The growth of the human population might be modelled by polynomials and exponential functions. Those models imply unbounded growth in the future. Concerning the worldwide population, however, taking into account widespread available contraception, enforced size of families, epidemic diseases, wars, and so on, unbounded population growth seems to be unrealistic even though the world population continues to increase alarmingly. More elaborate models that include adjustments to such exponential growth are available. Verhulst [61] proposed that a self-limiting process should operate when population becomes too large. For $N(t_0) = N_0$ he suggested

$$N(t) = \frac{N_0 K \exp(r(t - t_0))}{K + N_0(\exp(r(t - t_0)) - 1)},$$

where r and K are positive constants (compare (1.9) with a slightly different parameterization). This relation is called *logistic growth* in a population. Obviously, $N(t) \to K$ as $t \to \infty$. K is the *carrying capacity* of the environment, which is usually determined by the available sustaining resources, while r is a measure of the rate at which it is reached. For $N_0 < K/2$, the form of $N(t)$ has a typical sigmoid character, which is commonly observed in population data, compare Table 3.2 for the dynamics of the US population.

With initial guess

$$p^0 = (N_0, K, r)^0 = (1, 12, 0.021).$$

and $t_0 = 1900$, PTOL $= 10^{-12}$, the global Gauss-Newton method converges within 11 iteration steps (Fig. 3.5a). In Fig. 3.5b, we observe the typical *scissors*

Table 3.2 Real data (years 1700–2000) and hypothetic data (years 2050, 2100) for the US population

Year	1750	1820	1922	1960	1974	1987	2000	2050	2100
N	0.5	1	2	3	4	5	6.3	10	11.2

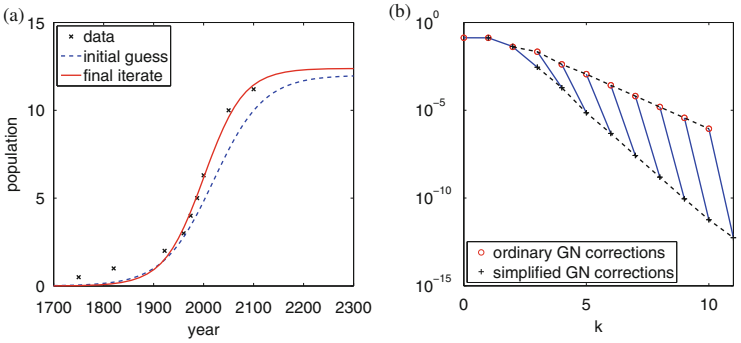


Fig. 3.5 Example 14. Verhulst population model. *Left*: Fit of model curve to data before and after parameter estimation. *Right*: Typical scissors between ordinary ($\sim\kappa$) and simplified ($\sim\kappa^2$) Gauss-Newton corrections as derived in (A.32). In this example one would get $\kappa = 0.24$

opening between ordinary and simplified Gauss-Newton corrections as stated in formulas (A.32).

A-Posteriori Perturbation Analysis

The above incompatibility factor has an interpretation in terms of *statistics*. In least squares problems, the minimization formulation is only reasonable, if small perturbations of the measured data lead to acceptable perturbations of the parameters, i.e., if the solution p^* is *stable* under small perturbations, say $\delta F(p^*)$. Then standard *linear* perturbation analysis yields the parameter perturbations

$$\delta p_L^* := -F'(p^*)^+ \delta F(p^*) \ .$$

For a *nonlinear* perturbation analysis, the convergence theory given in [6, 15] can be applied to yield

$$p^0 := p_{\text{old}}^* \ , \quad p_{\text{new}}^* := p_{\text{old}}^* + \delta p_{\text{NL}}^*$$

with the estimate

$$\delta p_{\text{NL}}^* \ \approx \ \frac{\delta p_{\text{L}}^*}{1 - \kappa(p^*)} \ . \tag{3.50}$$

Clearly, stability of the underlying statistical model can only be guaranteed for *adequate* nonlinear least squares problems. Note that this componentwise estimate may replace the normwise estimate (3.39).

Summary

In the frame of the Gauss-Newton or scoring method, a reliable numerical solution is required to pass the following three checks:

(a) *full rank condition* for the final Jacobian, defined via (3.18),
(b) *statistical well-posedness* via $\kappa(p^*) < 1$, see (3.37),
(c) a-posteriori perturbation analysis (3.50).

Experience as well as statistical analysis tells that whenever the ordinary Gauss-Newton method with *full rank* Jacobian fails to converge due to $\kappa(\cdot) \geq 1$, then one should improve the model.

Nonlinear Equality Constraints

There also exists a Gauss-Newton variant for nonlinear least squares problems with nonlinear equality constraints, which realizes a *QR*-factorization for linear equality constraints, see [15, Section 4.1.2]. In this case it is of utmost importance that the projectors $P(y)$ remain bounded so that all the algorithmic details can be easily translated from the unconstrained case to the constrained case. Interested readers may want to look up [15, Section 4.3].

Gauss-Newton Codes

The described global Gauss-Newton method with error oriented convergence criterion has been implemented in the codes NLSQ-ERR for unconstrained and NLSCON for equality constrained (including unconstrained) nonlinear least squares problems. For details, we refer the reader to [15].

3.4 Extension to ODE Models

At this point, we are now ready to put all pieces together that we need to identify parameters in systems biological models. These pieces include

- the automatic construction of parameter dependent *ODE models* from biochemical or physiological mechanisms (see Chap. 1, Sect. 1.2),
- *computational simulation* of the models by numerical integration (see Chap. 2, Sect. 2.5, for an educated choice of integrators), and
- *Gauss-Newton methods* for the arising nonlinear least squares problems (see this Chapter, Sects. 3.1–3.3), which, up to now, have only been presented for algebraic models.

In what follows, we will work out the modifications necessary for their application to ODE models as they arise in systems biology. The methods are implemented in the corresponding code BioPARKIN [24, 25].

3.4.1 Function Evaluation via Numerical Integration

Suppose we are given a system of ODEs

$$y' = f(y,p),\ y(0) = y_0, \quad y \in \mathbb{R}^d,\ p \in \mathbb{R}^q \tag{3.51}$$

that depends on unknown parameters $p = (p_1, \ldots, p_q)$. The aim is to identify these parameters such that the solution $y(t;p)$ matches given experimental data

$$(t_1, y_1, \delta y_1), \ldots, (t_M, y_M, \delta y_M),\ t_j \in \mathbb{R},\ y_j, \delta y_j \in \mathbb{R}^d\ ,$$

where $y_1, \ldots, y_M \in \mathbb{R}^d$ are the given *measurements* and $\delta y_1, \ldots, \delta y_M \in \mathbb{R}^d$ the corresponding *measurement tolerances*, see (3.2). Even though these tolerances are crucial for each parameter fitting problem, we will drop them in this section for ease of presentation. Quite often part of the components of $y, \delta y$ are missing, since no measurements are available; this case, too, we drop for ease of notation.

In order to generate the Gauss-Newton method, we first need to set up the *nonlinear least squares problem*

$$\|F(p)\|^2 = \min, \quad F(p) = \begin{pmatrix} y_1 - y(t_1,p) \\ \vdots \\ y_M - y(t_M,p) \end{pmatrix} \in \mathbb{R}^m, \quad m \le M \cdot d\ . \tag{3.52}$$

The computational evaluation of $y(t_1;p), \ldots, y(t_M;p)$ requires the numerical solution of the ODE (3.51), making sure that output data are available at the measurement points $t_1, \ldots, t_M$. As discussed at the end of Sect. 2.2.1 as well as in Sect. 2.5.1, one may need to apply the *dense output* mode of the selected numerical integrator. The numerical integration will be controlled by some *relative local error tolerance* TOL. Recall from Sect. 2.1.2 that only the local error can be controlled, but not the global error. Ignoring this difference for the moment, we may formally write that, instead of the *exact* mapping $F(p)$, we evaluate some *perturbed* mapping $\hat{F}(p)$ related to F in the form

$$\hat{F}(p) := F(p)(1+\sigma)\ , \quad |\sigma| \le \mathrm{TOL} \quad \text{or} \quad \|\hat{F}(p) - F(p)\| \le \|F(p)\|\ \mathrm{TOL} \tag{3.53}$$

Remark 11 The here advocated approach is also known as *single shooting* approach, since it involves only a continuous single trajectory $y(t), t \in [t_1, t_M]$.

Already in 1981, H. G. Bock [5, 6] had suggested and worked out a *multiple shooting* variant that works with a number of subtrajectories and additionally requires continuity conditions at the interfaces. This approach requires more array storage, which is prohibitive for large systems, but exhibits improved convergence properties with respect to the applied Gauss-Newton method. As the ODEs in systems biology are usually stiff, the single shooting approach will do.

3.4.2 Jacobian Approximation via Parameter Sensitivities

Apart from the mapping $F(p)$, we need a sufficiently accurate approximation of the Jacobian matrix $F'(p)$ to realize the Gauss-Newton method. Recall that, in contrast to Newton's method, which can compensate for a lack of Jacobian accuracy (e.g., in the "simplified" Newton method), the Gauss-Newton method requires more accuracy to solve the correct problem (which is why there is no "simplified" Gauss-Newton method, see Sect. 3.3.1). In our case the Jacobian matrix has the form

$$F'(p) = -\begin{pmatrix} s_1(t_1) & \dots & s_q(t_1) \\ \vdots & \vdots & \vdots \\ s_1(t_M) & \dots & s_q(t_M) \end{pmatrix},$$

wherein the elements $s_i(t_j)$ are the *parameter sensitivities* at the measurement points, as already introduced in Sect. 1.3.2. We here write them in terms of the (d, q)-matrix

$$S = (s_1, \dots, s_q) \quad \text{where} \quad s_i(t) = \frac{\partial y(t)}{\partial p_i} \in \mathbb{R}^d .$$

From (1.47) we know that these quantities satisfy the *parameter variational equations*

$$s_i' = f_y(y(t); p)s_i + f_{p_i}, \quad s_i(0) = 0 . \tag{3.54}$$

For the numerical approximation of the values $s_i(T)$, three options are in common use that we will no discuss in some detail. Recall from (3.53) that TOL is the relative accuracy required in the numerical integrator.

(a) *Direct numerical integration of compiled variational equations.* Assume some chemical compiler (see Sect. 1.2.3) has been used to implement f (say, by means of SBML); then it can be conveniently extended to generate the terms f_y, f_p automatically. Once the parameter variational equations have been explicitly generated, they can be solved numerically in a standard way. Recall that Eqs. (3.54) must be solved simultaneously with the original model equations $y' = f(y, p)$ to obtain the argument inside $f_y(\cdot), f_p(\cdot)$. This option is realized in

the software package BioPARKIN. It is usual to prescribe a relative integration error tolerance $\mathrm{TOL}_{\mathrm{sens}} > \mathrm{TOL}$.

(b) *Internal differentiation.* If no easy access to the analytical form of the derivatives f_y, f_p is at hand, these terms may be approximated by *internal* finite differences (see, e.g., H.G. Bock [5]), which means

$$f_y(\cdot) \approx \frac{f(y+\delta y, p) - f(y,p)}{\delta y}$$

for appropriately chosen terms δy. One may also realize the finite difference approximation via the chain rule as a directional derivative

$$f_y(y,p)y_p \approx \frac{f(y+\delta p\, y_p, p) - f(y,p)}{\delta p} \quad \text{for some "small"} \quad \delta p > 0.$$

This kind of implementation can be directly written inside the program for the evaluation of f, possibly also with the help of an automatic differentiation tool like ADOL, see, e.g., [30]. The choice of the deviation $\delta p > 0$ needs careful handling. Note that this approach does not need to implement the parameter variational equations, but requires extra effort to get the internal differentiation. In this option, the achieved relative error tolerance comes out to be $\mathrm{TOL}_{\mathrm{sens}} \sim \mathrm{TOL}$.

(c) *External numerical differentiation.* This option is most popular, since it does neither require the construction of the parameter variational equations nor an extra treatment like in the internal differentiation just above. Rather, it goes directly back to the definition of the parameter sensitivities (let $[0, T]$ denote the integration interval)

$$s_i(T) = \left.\frac{\partial y}{\partial p_i}\right|_T \approx \frac{y(T; p+\delta p_i) - y(T;p)}{\delta p_i}\,, \tag{3.55}$$

where the two terms in the numerator are just the numerical results from numerical integration of the ODEs for the parameter values $p + \delta p_i$ or p, respectively. Again, an efficient choice of the deviations δp_i requires subtle consideration. A detailed analysis including an adaptive δp-strategy is given in Appendix A.5. It shows that in this option the accuracy of the FD approximation of $F'(p)$ is at best $\mathrm{TOL}_{\mathrm{sens}} \sim \sqrt{\mathrm{TOL}}$, to be compared with the accuracy of the F-evaluation $\sim$ TOL. The accuracy change from TOL to $\sqrt{\mathrm{TOL}}$ indicates that the number of significant digits is *roughly halved*; for illustration, observe that we typically use $\mathrm{TOL} = 10^{-8}$ for the numerical integrators, which leads to $\sqrt{\mathrm{TOL}} = 10^{-4}$.

Despite its popularity, this option is less reliable in view of the Gauss-Newton method. Theoretically speaking, this lack of accuracy spoils the precise description of the subspace in the case of rank-deficiency. Practically speaking, the implemented Gauss-Newton iteration may in this case quite often deteriorate

to some erratic non-Gauss-Newton iteration – see, e.g., the tests presented in Sect. 3.5.3 below.

Remark 12 For the cut-off parameter ϵ_A needed for a Jacobian rank decision or a determination of the subcondition number of the Jacobian approximation (compare (3.18) in Sect. 3.2.2 and (A.13) in Appendix A.3) we thus obtain

$$\epsilon_A \sim \mathrm{TOL}_{\mathrm{sens}}, \quad A := F'(p), \quad \mathrm{sc}(A) \leq \mathrm{sc}_{\max} \sim 1/\mathrm{TOL}_{\mathrm{sens}} \,. \tag{3.56}$$

Parameter Transformation

For the sake of completeness, we finally mention the treatment of *inequality constraints*. In view of Table 3.1, let $p_i = \phi(u_i)$ denote some scalar mapping. In this case, the parameter sensitivities are calculated by the chain rule as

$$\frac{\partial y}{\partial u_i}(t) = \frac{\partial y}{\partial p_i}\frac{dp_i}{du_i} = s_i(t) \cdot \phi'(u_i) \,. \tag{3.57}$$

Differentiability After Discretization

Once approximations of the terms $F(p)$ and $F'(p)$ are available, the adaptive Gauss-Newton method can be realized as elaborated above in Sects. 3.3.1 and 3.3.2. However, as the mappings F and F' are obtained from discretization, subtle considerations are necessary to assure that sufficient differentiability is guaranteed. For further details and references see, e.g., [48].

3.4.3 Multiple Experiment Case

In many applications, the parameter identification problem at hand does not lead to a unique solution with full rank and $\kappa < 1$. In this situation, several experiments with different *internal* parameters may be coupled with the aim of achieving a common higher rank solution. This is called the "multiple experiment" case. Two types of examples are quite common:

(a) *different temperatures* $T_1, \ldots, T_n$: For this case, we refer to the modified Arrhenius law, which for selected parameters p states (dropping any indices)

$$p = A \cdot T^{\alpha} \cdot \exp(-\frac{\Delta E}{RT}) \,. \tag{3.58}$$

Herein A is the pre-exponential factor, ΔE the activation energy, T the temperature, and R the universal gas constant. The additional factor T^{α} makes the law "modified" in comparison with the usual Arrhenius law. As in Example 10, we

may take the logarithm of both sides of (3.58) and thus arrive at the different parameterization with new parameters (u_1, u_2, u_3) defined by virtue of

$$u = \ln p = \ln A + \alpha \ln T - \frac{\Delta E}{RT} =: u_1 + u_2 \ln T - \frac{u_3}{T} \ . \tag{3.59}$$

We remind the reader that this transformation also assures that the original parameters p remain positive throughout the Gauss-Newton iteration. From this we get

$$\frac{dp}{p} = du_1 + \ln T \, du_2 - \frac{1}{T} \, du_3 \tag{3.60}$$

and thus

$$\frac{\partial y}{\partial u} = s(t) \cdot p =: \bar{s}(t) \ .$$

Note that things simplify a lot, if the parameters p arise *linearly* in the reaction terms, see the argument against the explicit Michaelis-Menten kinetics (1.23) in Sect. 1.2.1 above. From this formula, we then obtain

$$\frac{\partial y}{\partial u_1} = \frac{\partial y}{\partial p} \frac{du}{du_1} = \bar{s}(t), \quad \frac{\partial y}{\partial u_2} = \bar{s}(t) \cdot \ln T, \quad \frac{\partial y}{\partial u_3} = \bar{s}(t)/T$$

which means that we merely need to compute the sensitivities $\bar{s}(t)$. The total Jacobian matrix has the block structure

$$F'(u) = \begin{pmatrix} F'(u;T_1), & F'(u;T_1) \cdot \ln T_1, & F'(u;T_1)/T_1 \\ \vdots & \vdots & \vdots \\ F'(u;T_n), & F'(u;T_n) \cdot \ln T_n, & F'(u;T_1)/T_n \end{pmatrix} ,$$

which shows that only the first block column needs to be computed, while the other two ones are obtained just by multiplication with a scalar factor. Note that (3.60) also indicates that the parameter A should be computed to *relative* accuracy, while the parameters α, E should be computed to *absolute* accuracy; such a treatment then results in relative accuracy for the parameter p.

(b) *different initial values* $y_0^1, \ldots, y_0^n$: In this case, the Jacobian matrix has the block structure

$$F'(p) = \begin{pmatrix} F'(p;y_0^1) \\ \vdots \\ F'(p;y_0^n) \end{pmatrix} \in \mathbb{R}^{mn \times q} \ ,$$

which means it is an (nm, q)-matrix. Such a structure would come up with any internal parameter.

3.5 Illustrative Examples

In order to illustrate different aspects of the quite subtle parameter identification task, we now add three illustrative examples of increasing complexity. Herein the first problem realizes a full-rank Gauss-Newton iteration, whereas the two further examples realize rank-deficient Gauss-Newton iterations.

3.5.1 Predator-Prey Model Revisited

In Sect. 1.1.2 above we had already introduced a model for two animal species, a predator and a prey. Here we take data from the Canadian lynx as predator and its primary prey, the snowshoe hare. Since it is impossible to count the exact number of hares in Canada in any given year, this information must be gained by capturing a small number of individuals and then estimating the actual number out in the wild. For over 300 years, the Hudson Bay Company has been involved in the fur trade in Canada. Detailed company records list the number of snowshoe hare pelts and the number of lynx pelts collected by hunters and trappers every year since the late 1700s. A small sample of this data is presented in Table 3.3. It is assumed that the numbers reflect a fixed portion of the total population of these animals. Although this assumption is of questionable accuracy, the data nevertheless represent one of the very few long term records available.

Let us revisit the model equations (1.10), which read

$$N_1' = N_1(\alpha - \beta N_2), \quad N_2' = -N_2(\gamma - \delta N_1).$$

As we have shown in Sect. 1.1.2, the solutions are closed orbits, which introduces the phase shift as an additional degree of freedom. With this insight in mind, we aim to compare two different strategies for parameter identification:

- *Case (P)*: We fix the initial values $N_1(0) = 30$ and $N_2(0) = 4$ according to the given data and estimate the model parameters α, β, γ and δ.
- *Case (P+I)*: We estimate the model parameters α, β, γ and δ together with the initial values $N_1(0)$ and $N_2(0)$.

In both cases, we choose the initial guesses

$$(\alpha^0, \beta^0, \gamma^0, \delta^0) = (0.5,\ 0.02,\ 1,\ 0.02)$$

to start the Gauss-Newton iteration. In case (P+I) we additionally choose

$$(N_1^0,\ N_2^0) = (30,\ 4).$$

For the required relative final accuracy of the iteration we set $PTOL = 10^{-4}$.

Table 3.3 *Predator-prey model.* Numbers of hares (prey) and lynxes (predator) in the years 1900–1920 recorded by the Hudson Bay Company

Year	Hares (in thousands)	Lynx (in thousands)
1900	30	4
1901	47.2	6.1
1902	70.2	9.8
1903	77.4	35.2
1904	36.3	59.4
1905	20.6	41.7
1906	18.1	19
1907	21.4	13
1908	22	8.3
1909	25.4	9.1
1910	27.1	7.4
1911	40.3	8
1912	57	12.3
1913	76.6	19.5
1914	52.3	45.7
1915	19.5	51.1
1916	11.2	29.7
1917	7.6	15.8
1918	14.6	9.7
1919	16.2	10.1
1920	24.7	8.6

Table 3.4 *Case (P)*: Iteration history for the global Gauss-Newton method

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	0.2721094e+02	0.352e-01		4
1	0.2695615e+02	0.351e-01	0.010	4
2	0.1326815e+02	0.469e-01	0.754	4
3	0.9471190e+01	0.443e-01	0.380	4
4	0.4933404e+01	0.161e-01	0.892	4
5	0.4239676e+01	0.451e-03	1.000	4
6	0.4236244e+01	0.298e-03	1.000	4
7	0.4236228e+01	0.911e-05	1.000	4
8	0.4236228e+01	0.482e-09	1.000	4

Case (P)

As documented in Table 3.4, the solution of the corresponding nonlinear least squares problem is obtained within 8 Gauss-Newton iterations, all of them with

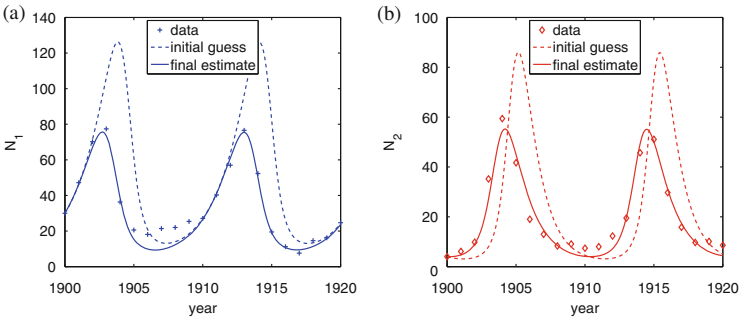


Fig. 3.6 *Case (P)*: Measurements and model simulations for initial guess and final Gauss-Newton iterate. *Left* (**a**): Prey population $N_1(t)$. *Right* (**b**): Predator population $N_2(t)$

Table 3.5 *Case (P+I)*: Iteration history for the global Gauss-Newton method

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	0.2721094e+02	0.201e+00		6
1	0.2695044e+02	0.201e+00	0.010	6
2	0.1037774e+02	0.120e+00	1.000	6
3	0.5679340e+01	0.195e+00	1.000	6
4	0.5020657e+01	0.144e+00	0.252	6
5	0.4206913e+01	0.127e-01	1.000	6
6	0.3767798e+01	0.150e-01	1.000	6
7	0.3763365e+01	0.200e-02	1.000	6
8	0.3763063e+01	0.177e-05	1.000	6

full rank 4. The final iterate came out as

$$p^* = (\alpha^*, \beta^*, \gamma^*, \delta^*) = (0.547534, 0.028119, 0.843175, 0.026558)\,.$$

The finally achieved accuracy was $2.87 \cdot 10^{-7}$, indicating that the parameter values have been determined up to 7 decimal digits. The incompatibility factor came out as $\kappa = 0.031$. For the final iterate the subcondition number turned out to be $\mathrm{sc}(F'(p^*)) = 90.9$, which confirms that the solution is unique. In Fig. 3.6, a comparison of measurements and model for both the initial guess and the final iterate is given.

Case (P+I)

As documented in Table 3.5, the solution of the corresponding nonlinear least squares problem is again obtained within 8 Gauss-Newton iterations, this time with

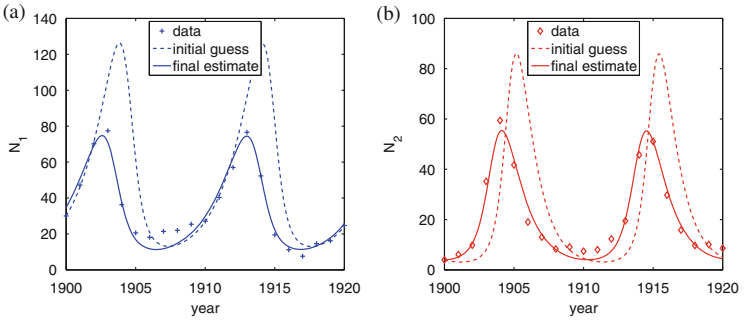


Fig. 3.7 *Case (P+I)*: Measurements and model simulations for initial guess and final Gauss-Newton iterate. *Left* (**a**): Prey population $N_1(t)$. *Right* (**b**): Predator population $N_2(t)$

full rank 6. The final iterate came out as

$$\begin{aligned} p^* &= (\alpha^*, \beta^*, \gamma^*, \delta^*, N_1(0)^*, N_2(0)^*) \\ &= (0.481599, 0.024847, 0.925125, 0.027508, 34.910103, 3.868005)\,. \end{aligned}$$

The finally achieved accuracy is $3.01 \cdot 10^{-4}$, indicating that the parameter values have been determined up to 4 decimal digits. The incompatibility factor came out as $\kappa = 0.13$, for the final iterate the subcondition turned out to be $\mathrm{sc}(F'(p^*)) = 400.0$, which here, too, confirms that the solution is unique. In fact, by leaving $N_1(0)$ and $N_2(0)$ open, a better fit than in case (P) is achieved. The least-squares functional was further reduced from $\|F(p^*)\| = 4.24$ in case (P) to $\|F(p^*)\| = 3.76$ in case (P+I). In the absence of any modelling error, this would point to a measurement error in the values $N_{1,2}(0)$, with $N_1(0) = 34.9$ thousands and $N_2(0) = 3.9$ thousands probably being the "true" values. However, model (1.10) represents a strong simplification. Certain mechanisms are missing, which is the main reason for the mismatch between data and simulation results. Nevertheless, the overall dynamical interactions between hares and lynxes are captured quite nicely by the model, as illustrated in Fig. 3.7.

3.5.2 A Simple Rank-Deficient Problem

Here we want to illustrate the importance of possible rank-deficiency in the course of the above presented Gauss-Newton iteration. For this purpose, we revisit the *bimolecular reaction* from Sect. 1.2.1, which reads

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C + D\,.$$

The corresponding ODE system (1.13) has been shown to be

$$A' = B' = -k_1AB + k_2CD, \quad C' = D' = +k_1AB - k_2CD \,,$$

wherein the rate coefficients k_1, k_2 have the role of the two parameters p_1, p_2 to be identified from a comparison with measurements. For initial values and rate coefficients we set

$$A(0) = 2,\ B(0) = 1,\ C(0) = 0.5,\ D(0) = 0, \quad k_1 = 0.1,\ k_2 = 0.2 \,.$$

From (1.15) we recall that in the chemical *equilibrium* phase classical *mass action kinetics* holds, which can be described by a *single* parameter, the Arrhenius coefficient

$$k_{21} = \frac{k_2}{k_1} = 2.0 \,. \tag{3.61}$$

With this insight in mind, we choose two different sets of measurements:

- *Case (T+E)*: measurement data cover both transient and equilibrium phase.
- *Case (E)*: measurement data cover equilibrium phase only.

In order to generate "measurements", we integrate the above ODE system by means of the explicit Runge-Kutta integrator DOPRI5 (see Sect. 2.2.1) with relative and absolute accuracies RTOL= 10^{-8}, ATOL= 10^{-6} and interpret the values computed for the adaptive time points as measurement data. In both cases, we choose the initial guesses

$$(k_1^0, k_2^0) = (0.2,\ 0.5)$$

to start the Gauss-Newton iteration. The required relative final accuracy of the iteration has been set to $PTOL = 10^{-4}$.

Case (T+E)

As documented in Table 3.6, the solution of the corresponding nonlinear least squares problem is obtained within 5 Gauss-Newton iterations, all of them with *full rank 2*. The final iterate came out as

$$p^* = (k_1^*, k_2^*) = (0.10000001, 0.19999997) \quad \Rightarrow \quad k_{21}^* = 1.9999994 \,.$$

Obviously, this recovers the original kinetic parameter values to 7 decimal digits in agreement with the output value of the final achieved accuracy $1.8 \cdot 10^{-7}$. In passing we note the (rounded) results for the incompatibility factor $\kappa = 0.008$ and for the subcondition number $\mathrm{sc}(F'(p^*)) = 6.3$ for the final iterate, which confirms

Table 3.6 *Case (T+E)*: Iteration history for global Gauss-Newton method

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	4.81e-02	3.55e-01		2
1	2.55e-02	1.75e-01	0.389	2
2	1.04e-02	4.22e-02	1.000	2
3	9.16e-04	1.90e-03	1.000	2
4	8.00e-06	1.99e-05	1.000	2
5	3.21e-07	1.54e-09	1.000	2

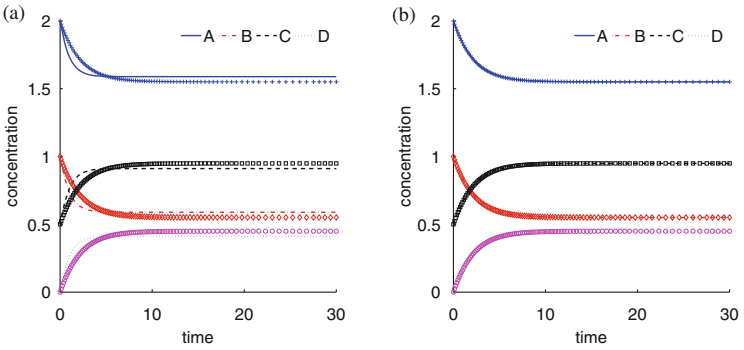


Fig. 3.8 *Case (T+E)*: Comparison of measurements and model. *Left* (**a**): Model simulation for initial guess (k_1^0, k_2^0). *Right* (**b**): Model simulation for final parameter (k_1^*, k_2^*)

Table 3.7 *Case (E)*: Iteration history for local Gauss-Newton method (within global GN algorithm). Due to the occurrence of rank-deficiency, a unique solution cannot be expected

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	3.85e-02	2.96e-02		1
1	2.40e-03	1.85e-03	1.000	1
2	1.11e-05	7.67e-06	1.000	1
3	6.67e-06	6.75e-11	1.000	1

the expectation of a unique solution. In Fig. 3.8, a comparison of measurements and model for both the initial guess and the final iterate is given.

Case (E)

As documented in Table 3.7, a solution of the corresponding nonlinear least squares problem is obtained within 3 local Gauss-Newton iterations, here, however, all of them with deficient *rank 1*. This means that we cannot expect to have obtained a unique solution.

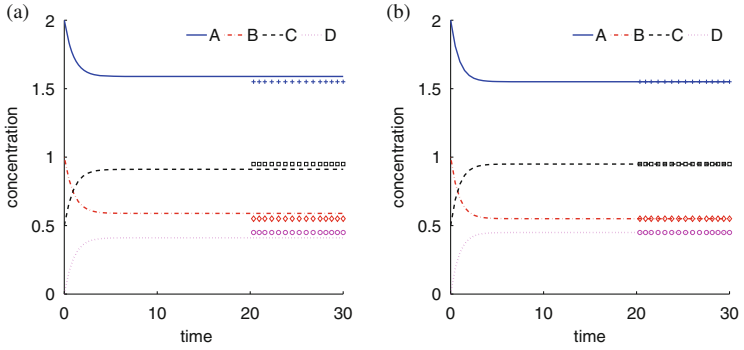


Fig. 3.9 *Case (E)*: Comparison of measurements and model. *Left* (**a**): Simulation of model for initial guess (k_1^0, k_2^0). *Right* (**b**): Simulation of model for final Gauss-Newton iterate (k_1^{**}, k_2^{**})

The final parameter iterate came out as

$$p^{**} = (k_1^{**}, k_2^{**}) = (0.24155702, 0.48312906) \quad \Rightarrow \quad k_{21}^{**} = 2.0000622\,.$$

Again, we note the rounded incompatibility factor (here for the rank-deficient Gauss Newton iteration) $\kappa = 0.004$. The final subcondition number (for full rank 2) came out to be $\mathrm{sc}(F'(p^*)) = 4.5 \times 10^6$, by the way only slightly less than the condition number $\mathrm{cond}\, F'(p^*) = 5.6 \times 10^6$ (computed here only for comparison reasons). In view of the numerical integration accuracy, the Jacobian accuracy value $\varepsilon_A = 10^{-4}$ has been set, compare Sect. 3.2.2. With this specification, the rule (3.18) gives rise to the observed rank deficiency. In Fig. 3.9, a comparison of measurements and model for both the initial guess and the final iterate is presented.

Clearly, the invested kinetic parameters $(k_1, k_2) = (0.1, 0.2)$ have *not* been recovered. Nevertheless, although the obtained result (k_1^{**}, k_2^{**}) is totally "off the track", the *rank-deficient* Gauss-Newton method is able (i) to deliver a "nice fit", see Fig. 3.9b, and (ii) to recover the Arrhenius coefficient k_{21}. The latter occurrence is in direct agreement with formula (3.30), as a careful examination would reveal (skipped here, too technical). However, it is important to understand that the obtained computational results cannot serve as interpretable values beyond the equilibrium phase.

Summary

For unaware observers it is certainly puzzling that both "solutions", $p^* = (k_1^*, k_2^*)$ as well as $p^{**} = (k_1^{**}, k_2^{**})$, yield the same "perfectly matching" plots compared with data set *(E)*. However, this is in agreement with the analysis given in (3.49): both solutions are minimal points in a parabolic "valley"

$$F(p) = F(p^*) + \mathcal{O}(\|p - p^*\|^2)\,, \quad F(p) = F(p^{**}) + \mathcal{O}(\|p - p^{**}\|^2)\,.$$

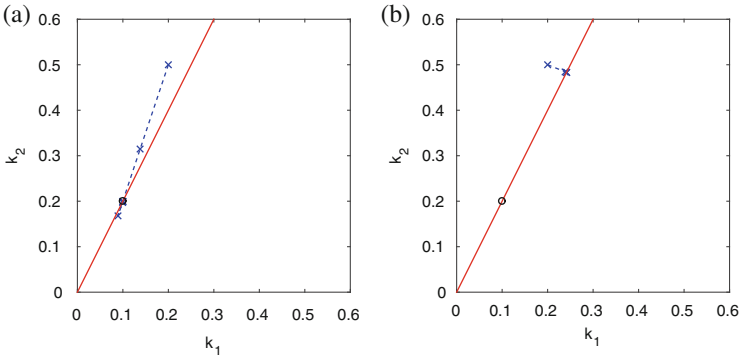


Fig. 3.10 Gauss-Newton iterations in the parameter plane. Unique solution point $p = (0.1, 0.2)$ marked as (o). Straight line represents subspace S according to (3.62). *Left* (**a**): Case (T+E). Full rank iteration $p^0 \to p^* = p$. Unique solution recovered. *Right* (**b**): Case (E). Rank deficient iteration $p^0 \to p^{**} \in S$. Observe the orthogonality of the iteration path towards the subspace

For further illustration, Fig. 3.10 shows the two cases in a (k_1, k_2)-plane. Starting from the same initial guess p^0, both final iterates p^* and p^{**} are contained in the subset

$$S := \{(k_1, k_2) \mid k_2 - 2.0\, k_1 = 0\}\,, \tag{3.62}$$

which was determined by the algorithm via the rank decision device within the $QR\hat{Q}$-decomposition of the Jacobian $F'(p)$. In passing, recall that, by algorithmic construction, the Gauss-Newton corrections are orthogonal to the nullspace of the Jacobians, which, assuming not too much Jacobian variation close to the solution point p^{**}, means also roughly orthogonal to S.

Remark 13 Note that the above occurrence in the rank-deficient case is in perfect agreement with the model reduction based on chemical insight as presented in Sect. 1.2.1 above. In [35] the question of identifiability of parameters in nonlinear dynamical models has also been treated in detail. In contrast to the presently advocated techniques, that article tries to implement an *interactive* technique that does not exploit any of the information coming from the Gauss-Newton method or its local realization via some rank-deficient QR-factorization. Here, however, the model reduction has been *automatically* realized by the algorithm itself.

3.5.3 A Complex Human Menstrual Cycle Problem

In this final section, we want to present more general parameter identification techniques assumed to be helpful when treating a wider class of complex problems. For this purpose, we revisit Example 1 (Sect. 1.2.1), which models the human

menstrual cycle. The underlying compartment model is shown in Fig. 1.6, a flowchart of part of the chemical reaction network is depicted in Fig. 1.7, a subset of ODEs for the hormone LH is written down directly after that figure. Details of the whole model have been published in [53]. The model leads to a system of 33 ODEs with 114 parameters, which is of moderate size compared to other applications, see, e.g., the model presented in Sect. 2.5.3. From the data given in [53], 63 degrees of freedom (in parameter space) had been identified using the techniques described in the present book. After the publication of that model with the identified parameters, new data became available from an EU project.[3] The problem to be presented here deals with the question of whether the published model would match the additional data, too. The notation follows the one in Sect. 1.2.1. All computations have been performed with the software package ZIB_RubyExt.

Before starting we want to emphasize that in this model *mass conservation* has not been implemented, so that in intermediate tests or iterations "blow-up" of numerical solutions may well occur (compare Remark 3 in Sect. 1.3.1 above).

Input Data

The new data were collected from 40 individual women.[4] Four different hormones were measured, namely LH (luteinizing hormone), FSH (follicle stimulating hormone), E2 (estradiol), and P4 (progesterone). Measurements took place roughly every second day. In order to be able to present the individual data in common, their LH peaks were aligned. In total, roughly 2500 data points arose. These so-called *raw data* can be seen in Fig. 3.13, left column.

In addition, we generated about 300 *averaged data* according to the rule

$$\bar{z}_i(t_j) := \frac{1}{N_{ij}} \sum_{n=1}^{N_{ij}} z_i^n(t_j) \ , \tag{3.63}$$

where N_{ij} denotes the number of individual measurements z_i^n of hormone i at day t_j. Once the averages have been calculated, the corresponding *standard deviations* were computed via the usual formula

$$\sigma_{z_i}^2(t_j) := \frac{1}{N_{ij}-1} \sum_{n=1}^{N_{ij}} \left(z_i^n(t_j) - \bar{z}_i(t_j)\right)^2 \ . \tag{3.64}$$

Both raw and averaged data only cover essentially one menstrual cycle, so they do not reflect any *periodicity*. In order to enforce a periodic solution, we would need

[3] PAEON: Model Driven Computation of Treatments for Infertility Related Endocrinological Diseases.

[4] Data courtesy due to Dorothea Wunder, CHUV Lausanne.

to solve a periodic boundary value problem, which, however, is beyond the scope of this book – see (1.6) in Sect. 1.1.1. As an ad-hoc strategy, we artificially expanded the new data by merely copying them for two or three more periods, each of them 28 days (as in the data from [53]). Such a strategy will only work, if the periodic orbits are dynamically stable, which we may assume in the physiological example at hand. So we expect a short transient phase that very soon ends up in some periodic motion.

Selection of "Most Sensitive" Parameters via Sensitivity Analysis

After establishing the ODE model, a typical second step in modelling is the computation of the parameter sensitivities $y_p(t)$, as described in Sect. 1.3.2, e.g., via formula (1.47) for the variational equations

$$y_p' = f_y(y(t), p)y_p + f_p, \quad y_p(0) = 0\ . \tag{3.65}$$

In the context of our present problem, we selected only the four measured hormones (LH, FSH, E2, P4) from the 33 components of y. As for the numerical integration of these ODEs, we experimentally selected a relative local error tolerance for the numerical integrators of TOL$= 10^{-8}$. Accuracies less stringent led to a less reliable behavior of the Gauss-Newton algorithm.

A rather popular procedure is to find out the "most sensitive" parameters with respect to the four species on the basis of a sensitivity analysis. As a prerequisite, one has to scale the elements of the sensitivity matrices (e.g., as suggested in [25]) according to

$$\left(y_p\right)_{i,j}|_{scal} = \frac{\partial y_i}{\partial p_j} \frac{\max\{|p_j|, thres(p_j)\}}{\max\{\max_t |y_i(t)|, thres(y_i)\}}\quad , \tag{3.66}$$

wherein *thres* are user specified threshold values for each associated element. In reality, we actually inspected all 114 sensitivities interactively, which is some task! From this set, the ones with the largest absolute scaled sensitivities were selected to yield the 13 "most sensitive" parameters. Luckily, in our case the selection generated the same set for all four hormones (which need not be in all cases!). In Fig. 3.11, the results for P4 are depicted together with the 13 SBML parameter IDs; for their biological interpretations see Table 3.8.

Generally speaking, sensitivities are a rather popular means to clear up the role of individual parameters within a model at hand. However, they do not contain any information about measurement points or measurement data. Nor do they indicate any linear dependencies among the parameters. As a consequence, the list of "most sensitive" parameters concluded from sensitivity analysis alone may be both too long and too short, as documented in Table 3.8 below. *This is reflected in the fact that the Gauss-Newton iterations for these 13 parameters failed to converge, both over raw and over averaged data.*

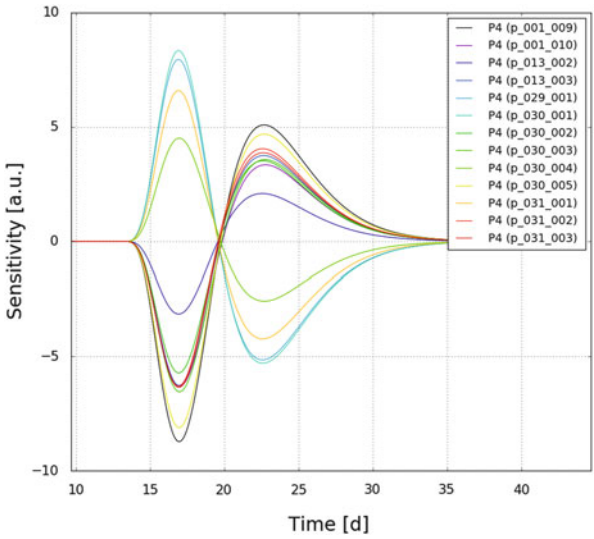


Fig. 3.11 Scaled parameter sensitivities for P4 (progesterone), one of the four measured quantities, with respect to a common set of 13 parameters selected as the "most sensitive" ones. Figure generated by BioPARKIN. The somewhat cryptic notation for the parameters is a consequence of the use of SBML, for their biological interpretation see Table 3.8

Table 3.8 List of 15 parameters selected as "most sensitive" ones. The first 13 parameters (between the first and second internal lines) were selected by sensitivity analysis, see Fig. 3.11. The five parameters with asterisks (*) were selected from *QR*-factorization, see Fig. 3.12; three of them already appear in the list from sensitivity analysis, whereas the two additional parameters (below the second internal line) are missing in the list derived from sensitivity analysis

ID	Name	Original value	Meaning
p_001_009*	$T^{\mathrm{LH}}_{\mathrm{G\text{-}R}}$	0.003 nmol/L	Threshold of GnRH on LH release
p_001_010	$n^{\mathrm{LH}}_{\mathrm{G\text{-}R}}$	5	Hill exponent
p_013_002*	$\mathrm{SF}_{\mathrm{LH\text{-}R}}$	2.726 IU/L	Scaling of LH receptor complex
p_013_003	$n^{\mathrm{AF3}}_{\mathrm{AF2}}$	3.689	Hill exponent (preferably integer!)
p_029_001	f_0	16/d	Mean frequency of GnRH pulse generator
p_030_001*	a_0	0.0056 nmol	Amount of GnRH released by one pulse
p_030_002	$T^{\mathrm{mass,1}}_{\mathrm{E2}}$	220 pg/mL	Threshold of E2 for stimulation of GnRH mass
p_030_003	$n^{\mathrm{mass,1}}_{\mathrm{E2}}$	2	Hill exponent
p_030_004	$T^{\mathrm{mass,2}}_{\mathrm{E2}}$	9.6 pg/mL	Threshold of E2 for inhibition of GnRH mass
p_030_005	$n^{\mathrm{mass,2}}_{\mathrm{E2}}$	1	Hill exponent
p_031_001	k^{G}_{on}	322.18 L/(d nmol)	Binding rate of GnRH to its receptor
p_031_002	k^{G}_{off}	644.35/d	Breakup rate of GnRH receptor complex
p_031_003	k^{G}_{degr}	0.447/d	Degradation rate of GnRH
p_001_001*	$b^{\mathrm{LH}}_{\mathrm{syn}}$	7310 IU/d	Basal LH synthesis rate constant
p_006_002*	T_{IhA}	95.81 IU/mL	Threshold of inhibin A in FSH synthesis

Selection of "Most Sensitive" Parameters via QR-Factorization of $F'(p^0)$

Having learned that sensitivity analysis is not enough to find the "most sensitive" parameters, we now turn to the parameter selection via the *QR*-factorization of

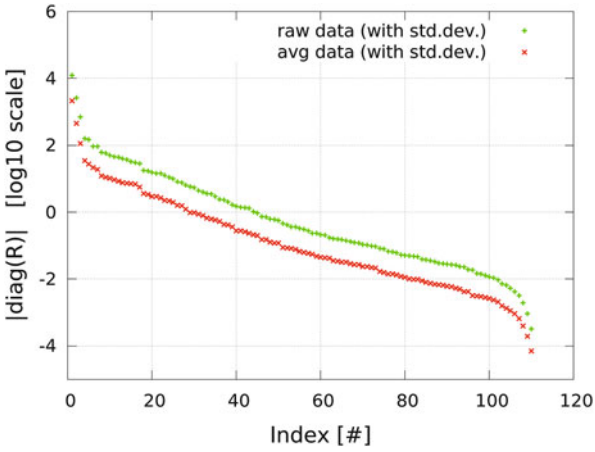


Fig. 3.12 Diagonal elements $|r_{11}|, |r_{22}| \ldots$, from QR-factorization of $F'(p^0)$ for both raw and averaged data. From the qualitative behavior, we selected the first five common parameters for both cases (see asterisks in Table 3.8)

the Jacobian matrix $F'(p^0)$, which also contains the sensitivities – however, now with row and column scaling according to (3.20), which is different from the above variant (3.66), but assures that the problem is statistically well-stated. In the present problem setting, the guess p^0 will naturally just be the original set of parameter values from the model in [53]. In Fig. 3.12, the absolute values of the diagonal elements, $|r_{11}|, |r_{22}|, \ldots$, from the QR-factorization are depicted based on sensitivities for both raw and averaged data. In both cases there are visible distinctions at ranks 3, 5 and 7 separating some subsets of "most sensitive" and "less sensitive" parameters. As a compromise between 3 (lower chance for obtaining a good fit) and 7 (higher computational costs) we decided to select 5 parameters. The thus selected five "most sensitive parameters" and their ranking are the same for both raw and averaged data. They are listed in Table 3.11; for their biological meaning and names we refer to Table 3.8. In a first attempt, we realized Gauss-Newton iterations for this parameter selection.

Identification of Five Preselected Parameters

In the Gaussian least squares problem formulation (3.2) we set the measurement tolerances equal to the standard deviation of the data as defined in (3.64),

$$\delta y_{ji} := \sigma_{z_i}(t_j) \ .$$

In what follows, we present the results of an identification of the five preselected parameters via Gauss-Newton methods, comparing the results for both raw and averaged data. For convenience and illustration purposes, the Jacobian approximation was realized via *external* differentiation as described in Sect. 3.4.2. In our computer runs, the choice of the relative deviation δp_i turned out to be really critical. We present the "best" runs, for a relative parameter accuracy PTOL$= 10^{-4}$ and

Table 3.9 *Raw data*: Convergence history for the identification of the five selected parameters. Final computational estimate of incompatibility factor $\kappa = 0.131$

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	0.5208387e+02	0.215e+00		5
1	0.5208275e+02	0.212e+00	0.010	5
2	0.5576308e+02	0.143e+00	1.000	5
3	0.5669474e+02	0.254e-01	1.000	5
4	0.5654622e+02	0.115e+00	0.189	5
⋮				
32	0.5179592e+02	0.257e-03	1.000	5
33	0.5180611e+02	0.216e-03	1.000	5
34	0.5180080e+02	0.559e-04	1.000	5

a deviation $\eta \cdot p_{\text{scal}}$ with $\eta = 10^{-5}$, larger values gave either bad or even no convergent results. This is in agreement with the warnings about the unreliable Jacobian accuracy obtained by the external differentiation as expressed in Sect. 3.4.2 above.

Raw data. In this setting we have 7635 points for the triple data set. The solution of the corresponding nonlinear least squares problem was obtained within 34 Gauss-Newton iteration steps. As can be (partly) seen in Table 3.9, full rank 5 occurred throughout the iteration. As at least two consecutive damping factors $\lambda = 1.00$ came out finally, an estimate for the incompatibility factor $\kappa = 0.131$ could be computed, see (A.32). As shown in Fig. 3.13, nearly no improvement was obtained, which might mean (in this reduced parameter subspace) that the parameters from [53] also match the new data. The finally achieved fit of model and data is presented in Fig. 3.13, left column. The fit as a whole looks quite nice.

Averaged data. As described above, we also generated average data, which led to 336 data points. The solution of the corresponding nonlinear least squares problem was obtained within 45 Gauss-Newton steps. A subset of the convergence history is given in Table 3.10. Throughout the iteration, full rank 5 occurred, just as in the case of the raw data. However, the computational estimate of the incompatibility factor appeared as $\kappa = 0.293$, quite different from the raw data value.

A comparison between the fit for the raw and the averaged data in Fig. 3.13 shows no significant differences between the two results, which is confirmed by similar final iterates listed in Table 3.11.

Parameter Identification over All Parameters

A more complete picture can be gained, if one is willing to spend more computing time. As mentioned above, the triple periodic set of averaged data gave rise to 336 data points. On this basis, we numerically solved 111 variational equations together with the original 33 parameter dependent ODEs, which resulted in $33 * (111 + 1) = 3696$ ODEs – obviously much more than in the attempts before with only 5 parameters. For the relative accuracy of the Jacobian matrix $A = F'(p)$ we applied

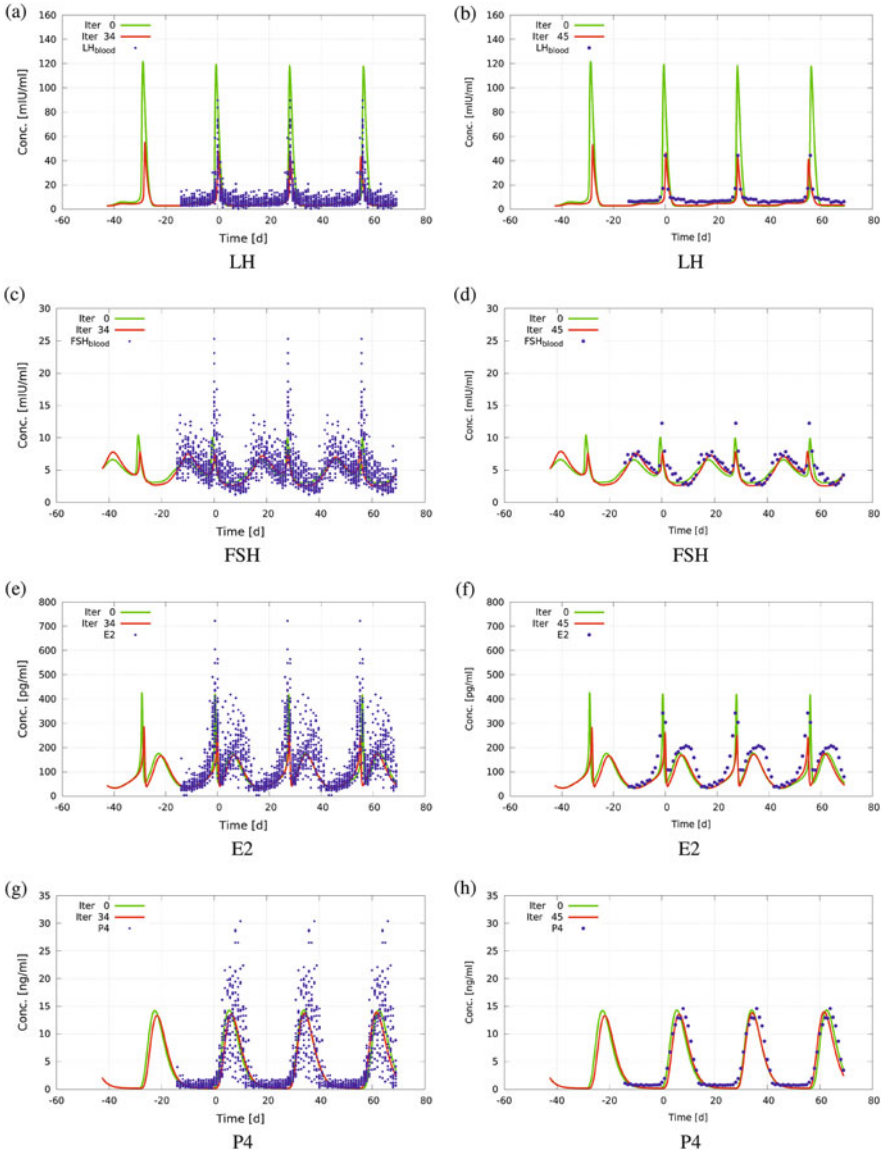


Fig. 3.13 *Raw data (left column) versus averaged data (right column) extended over three periods:* Gauss-Newton iteration over five parameters. Initial guess (*green*) versus final iterate (*red*) for the four measured hormones LH, FSH, E2, and P4

Table 3.10 *Averaged data*: Convergence history for the identification of the five selected parameters. Computational estimate of incompatibility factor appeared to be $\kappa = 0.293$

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	0.3437805e+02	0.204e+00		5
1	0.3438139e+02	0.202e+00	0.010	5
2	0.3843399e+02	0.953e-01	1.000	5
3	0.4046375e+02	0.870e-01	1.000	5
⋮				
43	0.3422895e+02	0.389e-03	1.000	5
44	0.3423697e+02	0.196e-03	1.000	5
45	0.3422478e+02	0.356e-04	1.000	5

Table 3.11 Parameter values in the first and final Gauss-Newton iterates for different runs (for the selection see the asterisks in Table 3.8). Note that the parameters differ by a factor of 10^7, which explains why scaling issues are important in the algorithmic realization

ID	Initial guess p^0 [53]	Final iterate p^* (raw data, 5 par.)	Final iterate p^* (avg. data, 5 par.)	Final iterate p^* (avg. data, 111 par.)
p_001_009	0.003	0.000413	0.000421	0.000239
p_013_002	2.726	1.979	1.958	2.003
p_030_001	0.0056	0.00686	0.00696	0.00438
p_001_001	7310	4984	4925	7708
p_006_002	95.81	91.59	92.44	72.85

the adaptive strategy (A.38) as worked out in Appendix A.5 based on the relative error tolerance $\mathrm{TOL} = 10^{-8}$ for the numerical integrator. Upon keeping in mind that there is an unknown constant $\mathcal{O}(1)$ involved, we set the threshold for the subcondition number

$$\mathrm{sc}(A) \leq \mathrm{sc}_{\max} := \frac{1}{\sqrt{10\ \mathrm{TOL}}} \approx 3200\,, \qquad .$$

In the absence of knowledge about the constant in (3.56), we tested several values of $\mathrm{sc}_{\max}$ between 1000 and 7000. The best results were obtained with $\mathrm{sc}_{\max} = 5500$, which are shown below in Fig. 3.14. For lower values of $\mathrm{sc}_{\max}$, the Gauss-Newton method did not converge (termination with $\lambda < 1$); for values above, the Gauss-Newton method converged in a subspace ($\lambda = 1$), but led to a poorer match.

The convergence history of the corresponding Gauss-Newton iteration is documented in Table 3.12; as can be seen, the choice of $\mathrm{sc}_{\max}$ led to an initial rank of only 10, while during the 42 iterations a maximum rank 41 appeared in-between as well as a final rank 20 with convergence in a stationary point. The estimated incompatibility factor in the 20-dimensional final subspace came out as $\kappa = 0.729$, which shows that the whole model is close to being incompatible already in this selected subspace.

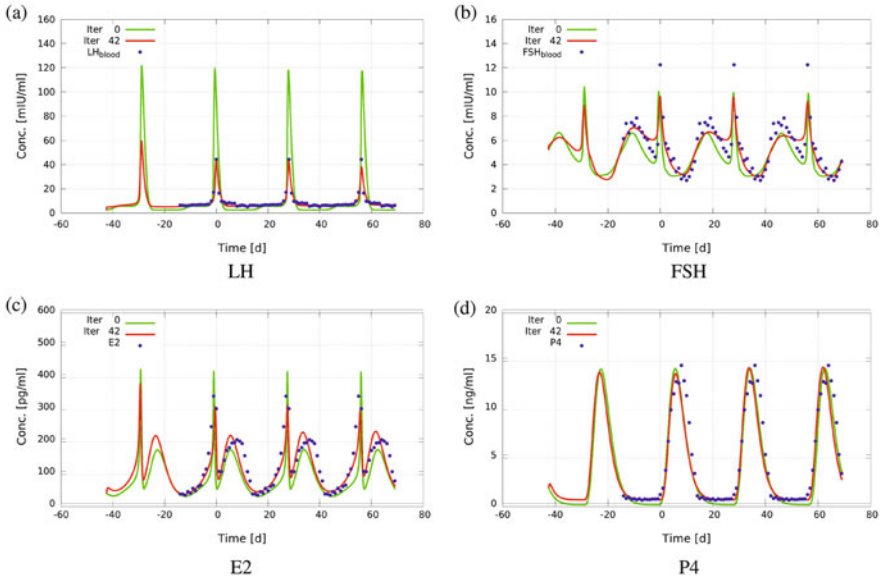


Fig. 3.14 *Averaged data over three periods:* Gauss-Newton iteration over all 111 parameters. Initial guess (*green*) versus final iterate (*red*) for the four measured hormones. Comparison of initial and final iterate: Note the better fits in LH, E2, P4 at the expense of a slightly worse fit in FSH; in particular, the rather satisfactory fit to the basal levels of LH and P4 is physiologically important

Table 3.12 Gauss-Newton convergence history over about 300 averaged data and 111 parameters. The computational estimate of the incompatibility factor appeared to be $\kappa = 0.729$

k	$\|F(p^k)\|$	$\|\Delta p^k\|$	λ_k	Rank
0	0.3437805e+02	0.218e-01		10
1	0.3432499e+02	0.216e-01	0.010	11
2	0.3405855e+02	0.175e-01	0.063	11
3	0.3390497e+02	0.114e-01	0.309	23
⋮				
37	0.2421932e+02	0.264e-01	0.422	41
38	0.2356051e+02	0.105e-01	0.904	36
39	0.2366233e+02	0.164e-01	0.446	36
40	0.2322512e+02	0.177e-01	0.382	29
41	0.2366426e+02	0.988e-02	1.000	20
42	0.2431996e+02	0.518e-02	1.000	20

A Appendix

A.1 Complex Exponential Function

Throughout the book, in particular in Sect. 2.1.3, the exponential function with complex argument plays an important role in connection with the Dahlquist test model (2.23). Let $z \in \mathbb{C}$ be a complex variable such that $z = a + \mathrm{i}b,\ a, b \in \mathbb{R}$ and i the imaginary unit with $\mathrm{i}^2 = -1$, then

$$|z|^2 = a^2 + b^2, \quad a = \Re z,\ b = \Im z$$

For the complex exponential function, the following notations are equivalent:

$$\exp(z) = e^z = e^{a+\mathrm{i}b} = e^a \cdot e^{\mathrm{i}b} \ .$$

For purely imaginary argument $\mathrm{i}b$, the famous Euler formula holds

$$e^{\mathrm{i}b} = \cos(b) + \mathrm{i}\sin(b) \quad \Rightarrow \quad |e^{\mathrm{i}b}|^2 = \cos^2(b) + \sin^2(b) = 1 \ . \tag{A.1}$$

From this, we immediately see that

$$|e^z| = e^a \begin{cases} < 1, & \text{if} \quad a < 0 \\ = 1, & \text{if} \quad a = 0 \\ > 1, & \text{if} \quad a > 0 \end{cases} \tag{A.2}$$

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0

In Sect. 2.1.3 we are mainly interested in the asymptotic properties of the complex exponential function for $z \to \infty$. This is rather different from the behavior with real arguments, as may be seen here:

$$e^z \overset{z\to\infty}{\longrightarrow} \begin{cases} 0, & \text{if} \quad a \to -\infty \\ [-1,1] + \mathrm{i}[-1,1], & \text{if} \quad a = 0,\ b \to \infty \\ \infty, & \text{if} \quad a \to \infty \end{cases} \tag{A.3}$$

Obviously, the limit depends on the path along which one approaches the complex point $z = \infty$, which is why in complex analysis this is called an *essential singularity*. In particular, observe that the path along the imaginary axis passes through *oscillations* with increasing frequencies b.

A.2 Condition Numbers of Linear Algebra Problems

Generally speaking, computational scientists (including computational systems biologists!) should be able to distinguish between the *condition of a problem* as opposed to the *stability of an algorithm*. For this reason, we recall some material from [17, chapter 2] here, which is of particular importance in connection with rank decision, see Sect. 3.2.2 and Appendix A.3.

Absolute and Relative Condition Number

Given some in general nonlinear mapping $r = f(e)$ evaluating the *result* r for given *input values* e. Then

$$\|f(\tilde{e}) - f(e)\| \dot{\leq} \kappa_{\text{abs}} \|\tilde{e} - e\| \quad \text{for} \quad \tilde{e} \to e$$

$$\frac{\|f(\tilde{e}) - f(e)\|}{\|f(e)\|} \dot{\leq} \kappa_{\text{rel}} \frac{\|\tilde{e} - e\|}{\|e\|} \quad \text{for} \quad \tilde{e} \to e$$

Whenever f is differentiable, we may thus write the *absolute* condition number as

$$\kappa_{\text{abs}} = \|f'(e)\|$$

and the *relative* condition number as

$$\kappa_{\text{rel}} = \frac{\|e\|}{\|f(e)\|} \|f'(e)\|.$$

For illustration, let us apply this definition to the simple case of a system of linear equations.

Condition of Linear Equation Systems

Let be given the linear system

$$Ap = y,\ A \in \mathbb{R}^{q\times q},\ \mathrm{rank}(A) = q.$$

Here input values are A, y. The mapping $p = f(A, y) = A^{-1}y$ describes the dependence of the result p on input data A and y. Application of the above definition for perturbations with respect to y then yields

$$\kappa_{\mathrm{abs}} = \|A^{-1}\|, \quad \kappa_{\mathrm{rel}} = \frac{\|y\|}{\|A^{-1}y\|}\|A^{-1}\| = \frac{\|Ap\|}{\|p\|}\|A^{-1}\| \le \|A\|\|A^{-1}\|$$

The same upper bound would arise for perturbations with respect to the matrix A, so that the *condition number*

$$\mathrm{cond}(A) := \|A\|\|A^{-1}\|, \qquad (\text{if } A \text{ is invertible}) \tag{A.4}$$

can be taken as a measure of the effect of perturbations of A, y to perturbations of the result p.

Example 15 We want to emphasize that the condition number of any problem is only half of the story. For instance, assume that you have to solve a linear equation $Ax = b$ in terms of the $(2, 2)$-matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & \epsilon \end{pmatrix}.$$

The condition number of this matrix is

$$\mathrm{cond}_2(A) \sim \frac{1}{\epsilon}.$$

Suppose now we have to solve the system for the two different right-hand sides

$$b_1 = (1, 0), \quad b_2 = (1, 1)$$

which would give rise to the two solutions

$$x_1 = (1, 0), \quad x_2 = \frac{1}{\epsilon}(\epsilon - 1, 1) \sim \frac{1}{\epsilon}(-1, 1).$$

Obviously, for $\epsilon \to 0$, the first solution remains unchanged, whereas the second one would blow up. In other words: The condition number describes the behavior of the second solution, but not that of the first one, i.e. it is merely a *worst case* quantity, which must be carefully analyzed in concrete situations.

Eigenvalues and Singular Values

In order to extend the condition number of a quadratic matrix to the case of a rectangular matrix, we need some facts about eigenvalues and singular values. For simplicity, we assume all matrices to be complex-valued.

- $\lambda \in \mathbb{C}$ is an eigenvalue of $A \in \mathbb{C}^{q\times q}$ if and only if there exists $x \in \mathbb{C}^q$ such that $Ax = \lambda x$.
- A^T has the same eigenvalues as A.
- λ^k is an eigenvalue of $A^k,\ k = 1, 2, \ldots$
- If A is invertible, then $\lambda(AA^T) = \lambda(A^TA)$.
- For any (m, q)-matrix A, there exists a decomposition of the kind

$$A = U\Sigma V^T, \quad \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)\ , \tag{A.5}$$

  where U, V are orthogonal matrices of proper dimension and $\sigma_1, \ldots, \sigma_q$ are the *singular values*.
- $\sigma_i(A) = \sqrt{\lambda_i(A^TA)} \geq 0$
- $\|A\|_2 = \sigma_1, \quad \|A^{-1}\|_2 = \sigma_q$, if A is nonsingular.

Condition Number of a Rectangular Matrix

Let $A \in \mathbb{R}^{m\times q}$, then

$$\mathrm{cond}(A) := \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|} = \frac{\sigma_1}{\sigma_q} \in [1, \infty]. \tag{A.6}$$

Assume now that $\mathrm{rank}(A) = q$. Then

$$\mathrm{cond}_2(A)^2 = \left(\frac{\sigma_1}{\sigma_q}\right)^2 = \frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)} = \mathrm{cond}_2(A^TA). \tag{A.7}$$

Condition of Linear Least Squares Problems

The argument based merely on the relation (3.11) is not precise enough. For a more careful analysis, we need the condition number of the full linear least squares problem, which we want to derive now. Assume (3.9) with $\mathrm{rank}(A) = q$. In order to

study the effect of perturbations of the right hand side y, the normal equations can be formally solved to yield the mapping

$$p = f(y) = (A^T A)^{-1} A^T y \, .$$

From this, we obtain the relative condition number (see above)

$$\begin{aligned} \kappa_{\text{rel},y} &= \frac{\|y\|_2}{\|p\|_2} \|f'(y)\|_2 = \frac{\|A\|_2 \|(A^T A)^{-1} A^T\|_2 \|y\|_2}{\|A\|_2 \|p\|_2} \\ &= \text{cond}_2(A) \frac{\|y\|_2}{\|A\|_2 \|p\|_2} \leq \text{cond}_2(A) \frac{\|y\|_2}{\|Ap\|_2} = \frac{\text{cond}_2(A)}{\cos\theta} \, , \end{aligned}$$

where we have used the definition of the angle θ from Fig. 3.3, right. In order to study perturbations with respect to A, we define the mapping

$$p = f(A) = (A^T A)^{-1} A^T y \, ,$$

from which, after some calculation (see, e.g., [17, Theorem 3.11]), we may derive the corresponding relative condition number

$$\kappa_{\text{rel},A} = \frac{\|A\|_2}{\|p\|_2} \|f'(A)\|_2 \leq \text{cond}_2(A) + \text{cond}_2(A)^2 \tan\theta \, .$$

In both condition numbers, attention will focus on the "large residual" case

$$\theta \to \pi/2 \quad \Rightarrow \quad 1 \ll \tan\theta \leq \frac{1}{\cos\theta} .$$

In this case, the algorithm via the normal equations is stable, but the problem itself is ill-conditioned! Hence, for principal reasons, the numerical solution of the normal equations will not be the right way to tackle linear least squares problems.

A.3 QR-Factorization with Column Pivoting

In order to solve linear least squares problems via QR-factorization, we are only left with determining appropriate orthogonal transformations.

Householder Transformations

For reasons to become apparent below, we select Householder matrices

$$H = I - \beta v v^T, \qquad \beta = 2/(v^T v) \, ,$$

wherein $v \in \mathbb{R}^d$ and dimension d can be chosen. For these (d,d)-matrices the following properties hold:

(a) H is *symmetric*, i.e. $H^T = H$,
(b) H is *orthogonal*, i.e. $HH^T = H^T H = I$,
(c) H is *scaling invariant*, i.e. $H(v) = H(\gamma v), \gamma \neq 0$

To start with, we show how to transform a given arbitrary vector $y \in \mathbb{R}^d$ to a multiple of the unit vector e_1, i.e.

$$Hy = y - 2\frac{v^T y}{v^T v} v = \alpha e_1 \quad \Longrightarrow \quad |\alpha| = \|y\|_2, \quad v = \gamma(y - \alpha e_1) \,, \tag{A.8}$$

where the γ is arbitrary, see property (c) above. Hence, we may set $\gamma = 1$ and get

$$v = y - \alpha e_1 = (y_1 - \alpha, y_2, \ldots, y_d)^T \,.$$

In order to avoid cancellation of leading digits in the first component, we dispose about the sign of α by setting $\alpha = -\operatorname{sign} y_1 \|y\|_2$. So we end up with a change of only the single component y_1 to v_1, while the other components remain unchanged and can just be left on the storage $y_2, \ldots, y_d$.

QR-Factorization

We now extend this procedure in such a way that the given (m,q)-matrix A is transformed into an upper triangular matrix $R = (r_{ij})$. We aim at a columnwise application, which means an application from the left:

$$HA = (I - \beta v v^T)A = A - \beta v w^T \quad \text{where} \quad w^T = v^T A$$

For step $k = 1, \ldots, q$, let $H_k = \operatorname{diag}(I_{k-1}, H'_k)$, where H'_k are Householder submatrices that only change the right lower corner matrices of the transformed matrices $A^{(k)}$. Then we obtain the following explicit pattern (merely represented by asterisks)

$$A^{(k)} = H_k \begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ & \ddots & & & \vdots \\ & & * & \cdots & * \\ & & \vdots & & \vdots \\ & & * & \cdots & * \end{pmatrix} = \begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ & \ddots & & & \vdots \\ & & * & \cdots & * \\ & & 0 & & \vdots \\ & & * & \cdots & * \end{pmatrix}$$

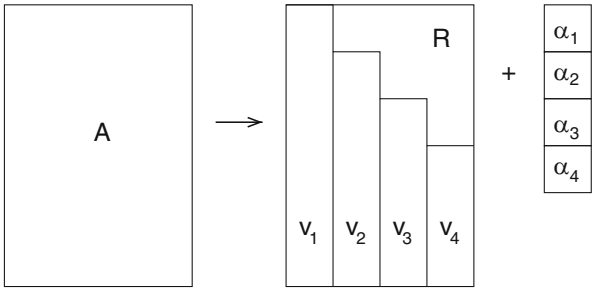


Fig. A.1 QR-factorization by product of Householder matrices: storage scheme

By comparison with (A.8) we see that each subcolumn k is transformed into some $\alpha_k e_1$ with a unit vector e_1 of the proper length. Hence, the diagonal elements of the matrix R arise as

$$r_{kk} = \alpha_k, \quad k = 1, \ldots, q \, .$$

After q steps we arrive at

$$Q^T A = (H_q H_{q-1} \cdots H_2 H_1) A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Note that the product $Q = H_1 \cdots H_q$ is orthogonal, but not symmetric, even though each factor matrix H is symmetric, since $Q^T = H_q \cdots H_1$.

In actual implementation, only the upper triangular matrix R, the successively shorter vectors $v_1, \ldots, v_q$ and the diagonal elements $\alpha_1, \ldots, \alpha_q$ need to be stored. This can be done on the space of the original matrix A, see Fig. A.1

Computational cost. For this QR-factorization algorithm, we have the following main computational tasks in each subcolumn k

(a) one matrix vector multiplication ($v^T A$),
(b) one outer product update (vw^T),
(c) one matrix subtraction,

which is $\sim 4mk$ operations. Over all columns $k = 1, \ldots, q$, this sums up to total computational costs of

$$\sim 2q^2 m \text{ operations.}$$

Compared with (3.10) for the normal equations algorithm, this is a factor of 4 more – but numerically stable!

Column Pivoting

Following a suggestion due to Businger/Golub [9], the QR-decomposition is slightly modified. Suppose we are in step k of the Householder transformation and have the partitioning

$$A^{(k-1)} = \begin{pmatrix} R & S \\ 0 & T^{(k)} \end{pmatrix} . \tag{A.9}$$

Let $T_j^{(k)}$ denote column $j \in \{k, \ldots, q\}$ of $T^{(k)}$. Now, select that column of $T^{(k)}$ with maximum 2-norm, i.e. let

$$\|T^{(k)}\|_\square := \max_j \|T_j^{(k)}\|$$

in terms of the special norm $\|\cdot\|_\square$ that is particularly suited for the here considered algorithm; next, shift this column into position k. After the shift, the following relation holds:

$$|r_{kk}| = \|T_k^{(k)}\| = \|T^{(k)}\|_\square . \tag{A.10}$$

Consequently, after application of this procedure for all k, the arising diagonal elements of R are ordered in such a way that

$$|r_{11}| \geq \ldots \geq |r_{qq}| \geq 0 . \tag{A.11}$$

For the algorithmic implementation of the possible reordering, merely a single additional integer vector of length q must be stored, see Fig. A.1. Formally speaking, we have to replace (3.13) by

$$A\Pi = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \Leftrightarrow \quad Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \Pi^T , \tag{A.12}$$

wherein Π denotes some *permutation matrix* satisfying

$$\Pi \, \Pi^T = \Pi^T \Pi = I_q .$$

Rank Decision

This column ordering device is the basis for a cheap rank decision for the given matrix A. With (A.10) this definition of rank deficiency has its algorithmic counterpart as

$$Q^T A \Pi = \begin{pmatrix} R & S \\ 0 & T^{(\rho+1)} \end{pmatrix}, \ \|T^{(\rho+1)}\|_\square \leq \varepsilon_A \, \|A\|_\square , \tag{A.13}$$

wherein the cut-off parameter ε_A, a measure of the relative accuracy of the matrix A, requires careful consideration. For further computation, the remainder matrix is set to zero, i.e. $T^{(\rho+1)} \to 0$. On this basis, one applies another orthogonal transformation, this time from the right side, to finally obtain

$$Q^T A \Pi \, \hat{Q}^T = \begin{pmatrix} \hat{R} & 0 \\ 0 & 0 \end{pmatrix}, \tag{A.14}$$

For details of the algorithm see Sect. 3.2.2.

A.4 Convergence of Newton and Gauss-Newton Methods

In systems biology, Gauss-Newton methods arise in the context of parameter identification, see Sect. 3.3. Their corresponding convergence theory is easier to understand, if the one for Newton's method is presented before. A mathematical convergence theory for both Newton and Gauss-Newton methods, especially derived in view of adaptive algorithms, may be found in the research monograph [15]. Here we merely give some brief account of the underlying ideas.

Newton Method

Consider a system of q *nonlinear equations* $F(p) = 0$ for q unknown variables $p_1, \ldots, p_q$ to be solved. Then the local Newton iteration reads

$$p^{k+1} - p^k = \Delta p^k = -F'(p^k)^{-1} F(p^k) \in \mathbb{R}^q, \quad k = 0, 1, \ldots \, . \tag{A.15}$$

Of course, we have to assume that the Jacobian (n, n)-matrices $F'(p)$ are nonsingular in some sufficiently large neighborhood of the solution point p^*. The question is to find out, under which conditions and at which rate this iteration converges. For linear problems, this iteration converges in one single step. For nonlinear problems, we have to define how to measure "nonlinearity". In a first step, some *Lipschitz condition* is imposed. Among the various ways to phrase such a condition, we pick out the following one:

$$\|F'(z)^{-1}(F'(y) - F'(p))(y - p)\| \le \omega \|y - p\|^2 \, . \tag{A.16}$$

The thus defined *Lipschitz constant* ω is independent of any affine transformation of the nonlinear mapping, say $F \to G = AF$, i.e. independent of any choice of a nonsingular matrix A, since

$$G'(z)^{-1}(G'(y) - G'(p)) = F'(z)^{-1} A^{-1} A (F'(y) - F'(p)) = F'(z)^{-1}(F'(y) - F'(p)).$$

That is why this Lipschitz condition is called "affine invariant".

In a second step, we recur to the *mean value theorem* for vector-valued functions, $F(p+h) - F(p) = h \int_0^1 F'(p+sh)\, ds$, which implies

$$F(y) - F(p) - F'(p)(y-p) = \int_{s=0}^{1} \left(F'(p+s(y-p)) - F'(p)\right)(y-p)\, ds\,. \tag{A.17}$$

On this basis we may conclude (for well-defined p, y, z)

$$\begin{aligned}
&\|F'(z)^{-1}(F(y) - F(p) - F'(p)(y-p))\| \\
&= \left\| \int_{s=0}^{1} F'(z)^{-1} \left(F'(p+s(y-p)) - F'(p)\right)(y-p) ds \right\| \\
&\leq \int_{s=0}^{1} \|F'(z)^{-1} \left(F'(p+s(y-p)) - F'(p)\right)(y-p) ds\| \\
&\leq \omega \|y-p\|^2 \int_{s=0}^{1} s ds = \frac{\omega}{2} \|y-p\|^2\,.
\end{aligned}$$

Upon identifying $y = z = p^k, p = p^{k-1}$ we finally arrive at the famous *quadratic convergence* result

$$\|p^{k+1} - p^k\| \leq \frac{\omega}{2} \|p^k - p^{k-1}\|^2\,. \tag{A.18}$$

Obviously, this method converges in the frame of this estimate, if

$$\frac{\omega}{2} \|p^k - p^{k-1}\| < 1 \quad \text{for} \quad k = 1, 2, \ldots\,.$$

For the initial iterates, this means that

$$\|p^1 - p^0\| = \|\Delta p^0\| < \frac{2}{\omega}\,. \tag{A.19}$$

We see that this assumption is always satisfied for linear problems, since there $\omega = 0$. For $\omega > 0$, we obtain

$$\|p^{k+1} - p^k\| < \|p^k - p^{k-1}\| < \ldots < \|p^1 - p^0\| < \frac{2}{\omega}\,.$$

Summarizing, the Newton method (A.15) converges *locally*, see (A.19), and *quadratically*, see (A.18), to a unique solution point p^*.

Two algorithmic questions arise in connection with the local Newton method: (a) How can I recognize that the iteration runs within the local convergence domain? (b) What should be done, if this is definitely not the case? The answer to these question is given by an extension to the global Newton method.

Globalization. In order to expand the domain of convergence of the local Newton method (A.15), a usual device is to realize some *damping* of the Newton corrections Δp^k in the following way:

$$p^{k+1} - p^k = \lambda_k \Delta p^k, \quad \Delta p^k = -F'(p^k)^{-1} F(p^k), \quad k = 0, 1, \ldots . \tag{A.20}$$

The damping factors $0 < \lambda_k \leq 1$ are selected by means of the affine invariant monotonicity criterion

$$\|\overline{\Delta p^{k+1}}\|_2 \leq \|\Delta p^k\|_2, \quad \text{where} \quad \overline{\Delta p^{k+1}} = -F'(p^k)^{-1} F(p^k + \lambda_k \Delta p^k) \,, \tag{A.21}$$

i.e. the *ordinary* Newton corrections Δp^k are compared with the *simplified* Newton corrections $\overline{\Delta p^{k+1}}$. By a subtle adaptive damping strategy, *global* convergence can be obtained under certain additional theoretical assumptions. Needless to say that the same techniques are also useful to assure that the local Newton method will stay in the convergence domain, as in this case the damping factors $\lambda_k = 1$ are selected by the algorithm. For more details, especially about a theoretically backed strategy for choosing the damping factors, readers are again referred to the textbook [15].

Gauss-Newton Method

Suppose now that we have to solve the *nonlinear least squares problem* $\|F(p)\|_2 = \min$. From (3.35), we recall the *local* Gauss-Newton iteration as

$$p^{k+1} - p^k = \Delta p^k = -F'(p^k)^+ F(p^k) \in \mathbb{R}^q, \quad k = 0, 1, \ldots \tag{A.22}$$

where $F'(p)^+$ denotes the Moore-Penrose pseudo-inverse of the possibly rank-deficient Jacobian (m, q)-matrix. In a first step, the above Lipschitz condition (A.16) is replaced by the generalized Lipschitz condition

$$\|F'(z)^+ (F'(y) - F'(p))(y - p)\| \leq \omega \|y - p\|^2 \quad \text{for} \quad y - p \in \mathcal{R}(F'(p)^+) \,. \tag{A.23}$$

The mean value theorem is still applicable and supplies the result (A.17) as above.

With these preparations, we again study the convergence of the iterates, where we use the notation of the generalized projectors

$$P(p) = F'(p)^+ F'(p), \quad \bar{P} := F'(p) F'(p)^+ \,. \tag{A.24}$$

and their orthogonal complements $P^\perp(p), \bar{P}^\perp(p)$:

$$\begin{aligned} \|p^{k+1} - p^k\| &= \|F'(p^k)^+ F(p^k)\| \\ &= \|F'(p^k)^+ \left(F(p^k) - F(p^{k-1}) - F'(p^{k-1})(p^k - p^{k-1})\right) \end{aligned}$$

$$
\begin{aligned}
&+F'(p^k)^+F(p^{k-1}) - F'(p^k)^+F'(p^{k-1})F'(p^{k-1})^+F(p^{k-1})\| \\
\leq\ &\|F'(p^k)^+\left(F(p^k) - F(p^{k-1}) - F'(p^{k-1})(p^k - p^{k-1})\right)\| \\
&+\|F'(p^k)^+\bar{P}^\perp(p^{k-1})F(p^{k-1})\|
\end{aligned}
$$

Obviously, the first term above can be treated as in the Newton case to yield

$$
\|F'(p^k)^+\left(F(p^k) - F(p^{k-1}) - F'(p^{k-1})(p^k - p^{k-1})\right)\| \leq \frac{\omega}{2}\|p^k - p^{k-1}\|^2 .
$$

The second term differs from the Newton case. Since $F'(p^{k-1})^+\bar{P}^\perp(p^{k-1}) = 0$, we may write

$$
\|F'(p^k)^+\bar{P}^\perp(p^{k-1})F(p^{k-1})\| = \|\left(F'(p^k)^+ - F'(p^{k-1})^+\right)\bar{P}^\perp(p^{k-1})F(p^{k-1})\|
$$

and are thus led to introduce the additional assumption

$$
\|F'(p^k)^+\bar{P}^\perp(p^{k-1})F(p^{k-1})\| \leq \kappa(p^{k-1})\|p^k - p^{k-1}\| \tag{A.25}
$$

for some *incompatibility factor* $\kappa(p)$. So we arrive at the convergence estimate

$$
\|p^{k+1} - p^k\| \leq \left(\frac{\omega}{2}\|p^k - p^{k-1}\| + \kappa(p^{k-1})\right)\|p^k - p^{k-1}\| . \tag{A.26}
$$

Note that this is no longer "quadratic" convergence. Obviously, the Gauss-Newton iterates converge in the frame of the estimate, if we assume that $\kappa(p) \leq \bar{\kappa} < 1$ so that

$$
\frac{\omega}{2}\|p^k - p^{k-1}\| + \kappa(p^{k-1}) \leq \frac{\omega}{2}\|p^k - p^{k-1}\| + \bar{\kappa} < 1
$$

can be achieved for $k = 1, 2, \ldots$. For the initial iterates, this means that we need to require

$$
\|p^1 - p^0\| = \|\Delta p^0\| < \frac{2(1-\bar{\kappa})}{\omega} . \tag{A.27}
$$

We see that this assumption is always satisfied for linear problems, since there we have $\omega = 0$ and $\bar{\kappa} = 0$. For the nonlinear case, we obtain

$$
\|p^{k+1} - p^k\| < \|p^k - p^{k-1}\| < \ldots < \|p^1 - p^0\| < \frac{2(1-\bar{\kappa})}{\omega} .
$$

Summarizing, the Gauss-Newton method (A.22) converges *locally*, see (A.27). However, in general its asymptotic convergence rate is not *quadratic* as in the ordinary Newton method, but *linear*, see (A.26), whenever a local solution point p^* exists. More precisely, this local convergence only holds for a restricted class of

nonlinear least squares problems. To identify this class, the following definition is helpful.

Definition *Adequate* nonlinear least-squares problems are characterized by the condition

$$\kappa(p^*) < 1 \,. \tag{A.28}$$

Note that this definition is more precise than the formerly used vague term "nearly compatible". In order to better understand the definition, let us study a few examples.

- *Linear least squares problems*: We have $F'(p) = A$, so that we obtain

  $$\omega = 0, \ \kappa(p) = \bar{\kappa} = 0 \,,$$

  which implies $\Delta p^2 = 0$ and therefore $p^1 = p^*$, i.e. convergence within *one* iteration.
- *Systems of nonlinear equations* ($m = q$): Let us assume that all Jacobian matrices are nonsingular so that we have $F'(p)^+ = F'(p)^{-1}$. Upon using the projection properties (3.28) we get

  $$\|F'(p)^+ \bar{P}^{\perp}(p) F(p)\| = 0 \quad \Rightarrow \quad \kappa(p) = 0 \,.$$

  Hence, we obtain *quadratic* convergence for the case of Newton's method.
- *Compatible nonlinear least squares problems*: Whenever $F(p^*) = 0$, we easily derive that

  $$\kappa(p^*) = 0 \,.$$

  In this case, under certain continuity assumptions, there is always some neighborhood of the solution point such that $\bar{\kappa} < 1$. Of course, we usually cannot compute $\kappa(p^*)$, but will find convenient estimates of $\kappa(p^k)$ in the course of the iteration, see (A.32) below.

For *incompatible* problems, we have to carefully observe the factor $\kappa(p)$ along the iterates, which is why it is called *incompatibility factor*: Whenever $\kappa(p^k) < 1$ is observed at all iterates p^k, then we assume that also $\kappa(p^*) < 1$ and therefore call the nonlinear least squares problem *adequate*. For *inadequate* problems with $\kappa(p^*) \geq 1$, H.G. Bock [6] proved that the point p^* is no longer a local minimum of $\|F(p)\|$.

In summary, the Gauss-Newton method converges locally linearly for adequate and locally quadratically for compatible nonlinear least-squares problems. Algorithmically, its convergence will be monitored within the corresponding globalization.

Globalization. In order to expand the domain of convergence of the local Gauss-Newton method (A.22), an adaptive *damping strategy* is applied as in Newton's method:

$$p^{k+1} - p^k = \lambda_k \Delta p^k, \quad \Delta p^k = -F'(p^k)^+ F(p^k), \quad k = 0, 1, \ldots \tag{A.29}$$

The damping factors $0 < \lambda_k \leq 1$ are chosen such that an "affine invariant" monotonicity criterion

$$\|\overline{\Delta p^{k+1}}\| \leq \|\Delta p^k\|, \quad \text{where} \quad \overline{\Delta p^{k+1}} = -F'(p^k)^+ F(p^k + \lambda_k \Delta p^k) \tag{A.30}$$

holds, wherein the *ordinary* Gauss-Newton corrections Δp^k are compared with the *simplified* Gauss-Newton corrections $\overline{\Delta p^{k+1}}$. Under additional mathematical assumptions *global* convergence can be achieved for *adequate* nonlinear least squares problems. As in Newton's method, the local Gauss-Newton iteration is monitored to stay in its convergence domain, whenever the damping factors λ_k remain to be selected as the value 1.

Computational characterization of adequateness. As soon as damping factors $\lambda_k = 1$ occur *at least twice consecutively before the final iterate* in the damping strategy [15], then *computational* estimates for the *theoretical* quantities ω and κ become available. For estimating the Lipschitz constant ω, the GN iteration can be further analyzed via

$$\|\overline{\Delta p^{k+1}}\| \leq \frac{\omega}{2} \|\Delta p^k\|^2 \quad \Rightarrow \quad \omega \geq \frac{2\|\overline{\Delta p^{k+1}}\|}{\|\Delta p^k\|^2} \approx \omega \tag{A.31}$$

For estimating the incompatibility factor κ, the convergence results (A.26) and (A.31) applied to indices $k, k+1$ yield

$$\kappa \approx \frac{\|\Delta p^{k+1}\|}{\|\Delta p^k\|}, \quad \kappa^2 \approx \frac{\|\overline{\Delta p^{k+1}}\|}{\|\overline{\Delta p^k}\|} \,. \tag{A.32}$$

An illustration of this feature is given in Fig. 3.5b. For a mathematical derivation of the above formulas interested readers are again referred to [15].

A.5 Adaptive External Numerical Differentiation

In this part we study the approximation quality of finite difference methods for the derivative $F'(p)$, where $p \in \mathbb{R}^q$ are parameters of an ODE model $y' = f(y, p)$. Assume we want to approximate some column i of the derivative

$$\frac{\partial F}{\partial p_i} \approx \frac{F(p + \eta_i \, p_{i,scal} \, e_i) - F(p)}{\eta_i} \,, \quad \eta_i \neq 0 \,, \tag{A.33}$$

where e_i denotes the coordinate unit vector, $p_{i,scal}$ the i-th component of the scaling vector p_{scal}, and η_i the common relative deviation for the whole column i. In order to simplify notation, we drop all indices and set $\eta \in \mathbb{R}$ in the following elementary analysis. For the estimate to follow, let us introduce a Jacobian Lipschitz condition (different from (A.17), but in a similar spirit):

$$\|(F'(\bar{p}) - F'(p))(\bar{p} - p)\| \leq \omega \|\bar{p} - p\|^2$$

With this assumption, we obtain an estimate for the *discretization error* with the *exact* F as

$$\|F(p+\eta) - F(p) - F'(p)\eta\| \leq \int_{s=0}^{1} \| \left(F'(p+s\eta) - F'(p)\right) \eta\| \, ds \leq \frac{\omega}{2}\eta^2 \, . \tag{A.34}$$

For the perturbed mapping $\hat{F}(p)$, however, we have to recall the *finite difference error* (3.53), here used in the form

$$\|\hat{F}(p) - F(p)\| \leq \|F(p)\| \, \mathrm{TOL} \tag{A.35}$$

in terms of the local discretization error tolerance TOL applied in the numerical integrator. Upon combining (A.34) and (A.35) and applying the triangle inequality twice, we arrive at

$$\|\hat{F}(p+\eta) - \hat{F}(p) - F'(p)\eta\| \leq \frac{\omega}{2}\eta^2 + 2\|F(p)\| \, \mathrm{TOL} \, .$$

For the approximation quality by finite differences we hence obtain

$$\hat{\Delta}(\eta) := \left\| \frac{\hat{F}(p+\eta) - \hat{F}(p)}{\eta} - F'(p) \right\| \leq \frac{\omega}{2}\eta + 2\,\frac{\|F(p)\| \, \mathrm{TOL}}{\eta}$$

Taking the derivative of the upper bound we arrive at the condition for the inner extremum

$$\frac{\omega}{2} - 2\,\frac{\|F(p)\| \, \mathrm{TOL}}{\eta^2} = 0 \, .$$

From this we get the "optimal" values

$$\eta_{\mathrm{opt}} = 2\sqrt{\frac{\|F(p)\|}{\omega}}\sqrt{\mathrm{TOL}} \, , \qquad \hat{\Delta}(\eta_{\mathrm{opt}}) \leq 2\sqrt{\omega\|F(p)\|}\sqrt{\mathrm{TOL}} \, . \tag{A.36}$$

The change from TOL in (A.35) to $\sqrt{\text{TOL}}$ in (A.36) indicates that the number of significant digits in the FD approximation of F' is *roughly halved*, compared to those in F.

Note that this optimality result can also be interpreted as an exact balance between the F-evaluation error (left below) and the discretization error for F' (right below), which would yield

$$2\,\|F(p)\|\,\text{TOL} = \frac{\omega}{2}\eta^2\,.$$

Geometrically speaking, the quantity ω represents some "curvature" around the point where the FD approximation is wanted.

Normwise Adaptive Strategy

With this insight, we now aim at finding an "adaptive choice" of the deviation parameter η based on computationally available quantities. Of course, the term $\hat{\Delta}(\eta)$ is computationally unavailable. But, following an idea in [17, Exercise 4.7], we try with the quantity

$$\kappa(\eta) := \frac{\|\hat{F}(p+\eta) - \hat{F}(p)\|}{\|F(p)\|}\,.$$

A short calculation using the above upper bounds again leads to

$$\kappa(\eta) \le 2\text{TOL} + \frac{\omega}{2\|F(p)\|}\eta^2 + \frac{\|F'(p)\|}{\|F(p)\|}\eta\,.$$

Upon using that $\text{TOL} \ll \eta \sim \sqrt{\text{TOL}}$ we obtain the first order estimate

$$\kappa(\eta) \doteq \frac{\|F'(p)\|}{\|F(p)\|}\eta =: c_1\eta \sim \sqrt{\text{TOL}}\,.$$

Hence, we may derive

$$\frac{\kappa(\eta)}{\kappa(\eta_{\text{opt}})} \doteq \frac{\eta}{\eta_{\text{opt}}}\,.$$

If we set the ad-hoc choice $\kappa(\eta_{\text{opt}}) = \sqrt{10\,\text{TOL}}$, as suggested in [17, Exercise 4.7], we obtain an estimate $[\eta_{\text{opt}}]$ from an actually applied η as

$$[\eta_{\text{opt}}] \doteq \frac{\kappa(\eta_{\text{opt}})}{\kappa(\eta)}\eta = \frac{\sqrt{10\,\text{TOL}}}{\kappa(\eta)}\eta \sim \sqrt{\text{TOL}}\,. \tag{A.37}$$

Componentwise Adaptive Strategy

In the algorithmic realization, we must take into account that the above derivation should be performed for *each* component $\hat{F}_i, i = 1, \ldots, M$, since the balance between the FD error and the discretization error will be needed for each component. As a consequence we would obtain computational estimates

$$[\eta_{\mathrm{opt},i}] := \frac{\sqrt{10\ \mathrm{TOL}}}{\kappa_i(\eta)}\eta \sim \sqrt{\mathrm{TOL}}\,, \quad i = 1, \ldots, M$$

wherein we have set

$$\kappa_i(\eta) := \frac{|\hat{F}_i(p+\eta) - \hat{F}_i(p)|}{\hat{F}_{i,scal}(p)}\,, \quad i = 1, \ldots, M$$

with

$$\hat{F}_{i,scal}(p) := \max(|\hat{F}_i(p+\eta)|, |\hat{F}_i(p)|) > 0\,.$$

In passing we note that the above term $\kappa_i(\eta)$ is invariant under rescaling of the component $\hat{F}_i$. Upon recalling that the above suggestions $[\eta_{\mathrm{opt},1}], \ldots, [\eta_{\mathrm{opt},M}]$ would arise for each column of the Jacobian separately, we see that we thus obtain a suggested full matrix of relative deviations of the same size $M \times q$ as the Jacobian matrix $F'(p)$. For reasons of implementation, however, we prefer only *one* suggestion per column, which means that we need to average somehow.

Among several averaging techniques considered in detail, the following one due to U. Nowak [47] turned out to be most robust. In this technique, the M values $\kappa_1(\eta), \ldots, \kappa_M(\eta)$ are replaced by the root mean square

$$\overline{\kappa}(\eta) := \sqrt{\frac{1}{M}\sum_{i=1}^{M}\kappa_i^2(\eta)}\,.$$

Insertion into the normwise form (A.37) then yields

$$[\eta_{\mathrm{opt}}] = \frac{\sqrt{10\ \mathrm{TOL}}}{\overline{\kappa}(\eta)}\,\eta \sim \sqrt{\mathrm{TOL}} \tag{A.38}$$

The above recursive formula is implemented in the code NLSCON, which is a subroutine within BioPARKIN.

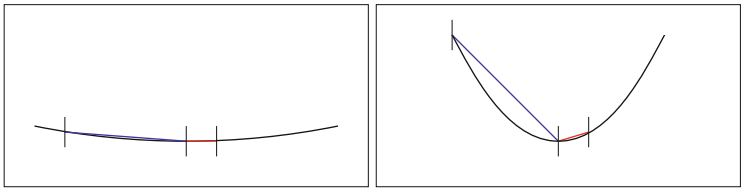


Fig. A.2 *Left:* "Small" curvature ω. Too large steps for the FD approximation still produce only small discretization error. *Right:* "Large" curvature ω. Too large steps for the FD approximation produce large discretization error

In order to understand why this formula is robust, let us look at a simple example: As shown in Fig. A.2, the discretization error is differently affected, if the curvature ω is either large or small.

Software

Numerical Algorithms

For the numerical algorithms described in this book there exist public domain software packages, which can be downloaded via internet. Here comes an incomplete (!) list including their present internet addresses:

Explicit Numerical Integrators for Non-stiff Initial Value Problems

DOPRI5, DOP853, ODEX Runge-Kutta codes based on formulas of Dormand/Prince and extrapolation code based on explicit midpoint rule, with control of step size (and order).
http://www.unige.ch/~hairer/software.html

DIFEX1 extrapolation code based on explicit midpoint rule, with order and step-size control.
http://www.zib.de/en/numerik/software/codelib/ivpode.html

LSODE Multistep codes based on Adams method with control of step size and order (slightly restricted to maintain quasi-uniform grids, if possible).
http://people.sc.fsu.edu/~jburkardt/f77_src/odepack/odepack.html

VODE Multistep codes based on Adams method with variable order and step-size control.
http://www.netlib.org/ode/vode.f

Implicit Numerical Integrators for Stiff Initial Value Problems

DASSL Multistep code based on the BDF discretization with control of order and step size.
http://www.netlib.org/ode/ddassl.f

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0

CVODE Solver for stiff and nonstiff ODE systems given in explicit form. The methods used in CVODE are variable-order, variable-step multistep methods. For nonstiff problems, CVODE includes the Adams-Moulton formulas, with the order varying between 1 and 12. For stiff problems, CVODE includes the Backward Differentiation Formulas (BDFs) in so-called fixed-leading coefficient form, with order varying between 1 and 5. For either choice of formula, the resulting nonlinear system is solved (approximately) at each integration step. CVODE is part of a software family called SUNDIALS.
http://computation.llnl.gov/casc/sundials/description/description.html

LSODI, LSODA Multistep codes based on BDF method with control of step size and order (slightly restricted to maintain quasi-uniform grids, if possible). **A** means **A**utomatic switching between the explicit Adams and the implicit BDF code.
http://people.sc.fsu.edu/~jburkardt/f77_src/odepack/odepack.html

RADAU5 implicit collocation code based on the Radau discretization with order and step-size control.
http://www.unige.ch/~hairer/software.html

Linearly Implicit Numerical Integrators for Stiff Initial Value Problems

LIMEX Extrapolation code based on linearly implicit Euler discretization with order and step-size control.
http://www.zib.de/en/numerik/software/codelib/ivpode.html

METAN1 Extrapolation code based on linearly implicit midpoint rule with order and step-size control.
http://www.zib.de/en/numerik/software/codelib/ivpode.html

RODAS, ROS3PL Rosenbrock-Wanner codes based on linearly implicit Runge-Kutta methods with step-size control.
http://www.unige.ch/~hairer/software.html

Numerical Integrators for Delay-Differential Equations

RETARD Explicit Runge-Kutta code, extension of DOPRI5 with step-size control.
http://www.unige.ch/~hairer/software.html

RADAR5 Implicit Runge-Kutta code, an extension of RADAU5 with step-size control.
http://www.unige.ch/~hairer/software.html

Parameter Identification Algorithms

NLSCON A Gauss-Newton method for nonlinear least squares problems with nonlinear equality constraints.
http://www.zib.de/en/numerik/software/codelib/nonlin.html

Software Packages

The task of numerical modelling in system biology will be of increasing complexity in the time to come. Therefore a number of elaborate software packages have been developed in recent years to assist the modelling process.

SBML Systems Biology Markup Language [11, 46].
http://www.sbml.org/

Copasi This is a software environment for "simulation and analysis of biochemical networks and their dynamics" [39]. For the numerical simulation, they suggest LSODA. For parameter identification, they provide a list of optimization routines, but none that analyzes the statistical background, as comparable to our Chap. 3.
http://www.copasi.org/

BioPARKIN (for: **Bio**logy-related **PAR**amater identification in large **KIN**etic networks) A software package for parameter identification problems in systems biology. This code can load models in the SBML format, which has become the standard in systems biology. Model construction in terms of adding or deleting ODEs is not (yet) possible in BioPARKIN. For this purpose, there exist numerous well established software tools, e.g., CellDesigner, see: http://www.celldesigner.org/. However, the user can edit mathematical expressions for kinetic rate laws, as well as numerical values for parameters and initial values. Numerical integration of the model system is standardly performed with LIMEX (see above), since this code especially offers advantages in connection with the solution of the variational equations for parameter sensitivities. Parameter identification is performed with the Gauss-Newton code NLSCON (see above). The code offers several unique features that are especially useful for biological modelling, such as breakpoint handling, or identifiability statements. The package is publicly available for download under

https://github.com/CSB-at-ZIB/BioPARKIN.

Further information, including a brief tutorial, can be found under

http://bioparkin.zib.de.

ZIB_RubyExt A software package for parameter identification problems in ODE systems. This program wraps the FORTRAN routines NLSCON and LIMEX to Ruby, a dynamic, open source programming language.

https://github.com/CSB-at-ZIB/ZIB_RubyExt
https://www.ruby-lang.org/en/

RoadRunner RoadRunner is a .NET library for carrying out deterministic simulation of SBML models. The library has been extensively tested and is available for all major operating systems. Apart from a .NET core, RoadRunner also relies on native libraries for: SBML support, conservation analysis, integration and steady state analysis.
http://roadrunner.sourceforge.net/RoadRunner/Welcome.html

libRoadRunner A high performance and portable simulation engine for systems and synthetic biology, libRoadRunner can run on many platforms including Windows, Mac OS, and Linux. libRoadRunner is a major rewrite of the original C# roadRunner by Bergmann and Sauro. The same original functionality however remains, including the original C API, the structural analysis code, sensitivity and steady state analyses, but with significant improvements to performance, back-end design, better event handling, new C++ API and stochastic simulation support. The use of LLVM (Low Level Virtual Machine), designed for real-time optimization and dynamic compilation of application software, allows for very fast simulations.
http://libroadrunner.org/

References

1. Amestoy, P., Duff, I., Koster, J., L'Excellent, J.Y.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM J. Matrix Anal. Appl. **23**(1), 15–41 (2001)
2. Amestoy, P.R., Buttari, A., Duff, I.S., Guermouche, A., L'Excellent, J.Y., Uçar, B.: MUMPS. In: Padua, D. (ed.) Encyclopedia of Parallel Computing. Springer, New York (2011)
3. Bader, G., Deuflhard, P.: A semi-implicit mid-point rule for stiff systems of ordinary differential equations. Numer. Math. **41**, 373–398 (1983)
4. Bader, G., Nowak, U., Deuflhard, P.: An advanced simulation package for large chemical reaction systems. In: Aiken, R.C. (ed.) Stiff Computation, pp. 255–264. Oxford University Press, New York/Oxford (1985)
5. Bock, H.G.: Numerical treatment of inverse problems in chemical reaction kinetics. In: Ebert, K.H., Deuflhard, P., Jäger, W. (eds.) Modelling of Chemical Reaction Systems, pp. 102–125. Springer, Berlin/Heidelberg/New York (1981)
6. Bock, H.G.: Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen. Ph.D. thesis, Universität zu Bonn (1985)
7. Boer, H.M.T., Stötzel, C., Röblitz, S., Deuflhard, P., Veerkamp, R.F., Woelders, H.: A simple mathematical model of the bovine estrous cycle: follicle development and endocrine interactions. J. Theor. Biol. **278**, 20–31 (2011)
8. Brown, P.N., Byrne, G.D., Hindmarsh, A.C.: VODE: a variable-coefficient ODE solver. SIAM J. Sci. Stat. Comput. **10**, 1038–1051 (1989)
9. Businger, P., Golub, G.H.: Linear least squares solutions by Householder transformations. Numer. Math. **7**, 269–276 (1965)
10. Butcher, J.C.: Coefficients for the study of Runge-Kutta integration processes. J. Aust. Math. Soc. **3**, 185–201 (1963)
11. Cornish-Bowden, A.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics **19**, 524–531 (2003)
12. Dahlquist, G.: Convergence and stability in the numerical integration of ordinary differential equations. Math. Scand. **4**, 33–53 (1956)
13. Deuflhard, P.: Order and stepsize control in extrapolation methods. Numer. Math. **41**, 399–422 (1983)
14. Deuflhard, P.: Recent progress in extrapolation methods for ordinary differential equations. SIAM Rev. **27**, 505–535 (1985)
15. Deuflhard, P.: Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms. Springer International, Heidelberg, New York (2002)
16. Deuflhard, P., Bornemann, F.: Scientific Computing with Ordinary Differential Equations. Texts in Applied Mathematics, vol. 42. Springer, New York (2002)

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0

17. Deuflhard, P., Hohmann, A.: Numerical Analysis in Modern Scientific Computing: An Introduction. Texts in Applied Mathematics, vol. 43, 2nd edn. Springer, New York (2003)
18. Deuflhard, P., Nowak, U.: Efficient numerical simulation and identification of large chemical reaction systems. Ber. Bunsenges **90**, 940–946 (1986)
19. Deuflhard, P., Nowak, U.: Extrapolation integrators for quasilinear implicit ODEs. In: Deuflhard, P., Engquist, B. (eds.) Large Scale Scientific Computing, pp. 37–50. Birkhäuser, Boston/Basel/Stuttgart (1987)
20. Deuflhard, P., Sautter, W.: On rank-deficient pseudoinverses. Lin. Alg. Appl. **29**, 91–111 (1980)
21. Deuflhard, P., Schütte, C.: Molecular conformation dynamics and computational drug design. In: Hill, J., Moore, R. (eds.) Applied Mathematics Entering the 21st Century. Invited Talks from the ICIAM 2003 Congress, pp. 91–119. SIAM, Philadelphia (2004)
22. Deuflhard, P., Bader, G., Nowak, U.: LARKIN—a software package for the numerical simulation of LARge systems arising in chemical reaction KINetics. In: Ebert, K.H., Deuflhard, P., Jäger, W. (eds.) Modelling of Chemical Reaction Systems, pp. 38–55. Springer, Berlin/Heidelberg/New York (1981)
23. Deuflhard, P., Hairer, E., Zugck, J.: One–step and extrapolation methods for differential–algebraic systems. Numer. Math. **51**, 501–516 (1987)
24. Dierkes, T., Wade, M., Nowak, U., Röblitz, S.: BioPARKIN – biology-related parameter identification in large kinetic networks. ZIB-Report 11–15, Zuse Institute Berlin (ZIB) (2011). http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1270
25. Dierkes, T., Röblitz, S., Wade, M., Deuflhard, P.: Parameter identification in large kinetic networks with BioPARKIN. arXiv:1303.4928 (2013)
26. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**, 19–26 (1980)
27. Ehle, B.L.: On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems. Research Report CSRR 2010, Department of AACS, University of Waterloo, Ontario (1969)
28. Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, Englewood Cliffs (1971)
29. Gragg, W.B.: Repeated extrapolation to the limit in the numerical solution of ordinary differential equations. Ph.D. thesis, University of California, San Diego (1963)
30. Griewank, A., Corliss, G.F. (eds.): Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia (1991)
31. Guglielmi, N., Hairer, E.: Implementing Radau II-A methods for stiff delay differential equations. Computing **67**, 1–12 (2001)
32. Hairer, E., Ostermann, A.: Dense output for extrapolation methods. Numer. Math. **58**, 419–439 (1990)
33. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, 2nd edn. Springer, Berlin/Heidelberg/New York (1996)
34. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I. Nonstiff Problems, 2nd edn. Springer, Berlin/Heidelberg/New York (1993)
35. Hengl, S., Kreutz, C., Timmer, J., Maiwald, T.: Data-based identifiability analysis on nonlinear dynamical models. Bioinformatics **23**, 2612–2618 (2007)
36. Hindmarsh, A.C.: LSODE and LSODI, two new initial value ordinary differential equations solvers. ACM SIGNUM Newsl. **15**, 10–11 (1980)
37. Hindmarsh, A.C., Serban, R.: User documentation for cvode v2.7.0. Technical Report UCRL-SM-208108, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory (2012)
38. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. **31**(3), 363–396 (2005)
39. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: COPASI – a COmplex PAthway SImulator. Bioinformatics **22**, 3067–3074 (2006)

40. Jones, D.S., Plank, M.J., Sleeman, B.D.: Differential Equations and Mathematical Biology. Mathematical and Computational Biology, 2nd edn. Chapman & Hall/CRC, Boca Raton (2010)
41. Kee, R.J., Miller, J.A., Jefferson, T.H.: CHEMKIN: a general-purpose, problem-independent, transportable, FORTRAN chemical kinetics code package. Technical Report SAND 80–8003, Sandia National Laboratory, Livermore (1980)
42. König, M., Holzhütter, H.G., Berndt, N.: Metabolic gradients as key regulators in zonation of tumor energy metabolism: a tissue-scale model-based study. Biotechnol. J. **8**, 1058–1069 (2013)
43. Lang, J., Teleaga, D.: Towards a fully space-time adaptive FEM for magnetoquasistatics. IEEE Trans. Magn. **44**(6), 1238–1241 (2008)
44. Maly, T., Petzold, L.: Numerical methods and software for sensitivity analysis of differential-algebraic systems. Appl. Numer. Math. **20**, 57–79 (1996)
45. Murray, J.D.: Mathematical Biology I: An Introduction. Interdisciplinary Applied Mathematics, vol. 17, 3rd edn. Springer, Heidelberg, New York (2008)
46. Novère, N.L., et al.: Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Res. **34**, D689–D691 (2006)
47. Nowak, U.: Adaptive finite difference approximation of Jacobian matrices. private communication, software NLSCON (1991)
48. Nowak, U., Deuflhard, P.: Numerical identification of selected rate constants in large chemical reaction systems. Appl. Numer. Math. **1**, 59–75 (1985)
49. Penrose, R.: A generalized inverse for matrices. Proc. Camb. Philos. Soc. **51**, 406–413 (1955)
50. Peters, G., Wilkinson, J.: The least squares problem and pseudoinverses. Comput. J. **13**, 309–316 (1970)
51. Petzold, L.R.: A description of DASSL: a differential/algebraic system solver. In: Scientific Computing, pp. 65–68. North-Holland, Amsterdam/New York/London (1982)
52. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. (eds.): Numerical Recipes in Fortran 77, 2nd edn. Cambridge University Press, Cambridge (1992)
53. Röblitz, S., Stötzel, C., Deuflhard, P., Jones, H., Azulay, D.O., van der Graaf, P., Martin, S.: A mathematical model of the human menstrual cycle for the administration of GnRH analogues. J. Theor. Biol. **321**, 8–27 (2013)
54. Russell, R.D., Shampine, L.: A collocation method for boundary value problems. NM **19**, 1–28 (1972)
55. Schlegel, M., Marquardt, W., Ehrig, R., Nowak, U.: Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation. Appl. Numer. Math. **48**(1), 83–102 (2004)
56. Shampine, L.F., Thompson, S.: Solving DDEs in MATLAB. Appl. Numer. Math. **37**, 441–458 (2001)
57. Sidje, R.B.: Expokit: a software package for computing matrix exponentials. ACM Trans. Math. Softw. **24**, 130–156 (1998)
58. Stötzel, C., Plöntzke, J., Heuwieser, W., Röblitz, S.: Advances in modeling of the bovine estrous cycle: synchronization with pgf2α. Theriogenology **78**(7), 1415–1428 (2012)
59. Stuart, A.M.: Inverse problem: a Bayesian perspective. Acta Numer. **19**, 451–559 (2010)
60. Vanlier, J., Tiemann, C.A., Hilbers, P.A.J., van Riel, N.A.W.: Parameter uncertainty in biochemical models described by ordinary differential equations. Math. Biosci. **246**, 305–314 (2013)
61. Verhulst, P.F.: Notice sur la loi que la population suit dans son accroissement. Corr. Math. et Phys. **10**, 113–121 (1838)
62. Widlund, O.: A note on unconditionally stable linear multistep methods. BIT **17**, 65–70 (1967)

Index

P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*, Texts in Computational Science and Engineering 12, DOI 10.1007/978-3-319-20059-0

Editorial Policy

1. Textbooks on topics in the field of computational science and engineering will be considered. They should be written for courses in CSE education. Both graduate and undergraduate textbooks will be published in TCSE. Multidisciplinary topics and multidisciplinary teams of authors are especially welcome.

2. Format: Only works in English will be considered. For evaluation purposes, manuscripts may be submitted in print or electronic form, in the latter case, preferably as pdf- or zipped ps-files. Authors are requested to use the LaTeX style files available from Springer at: http://www.springer.com/authors/book+authors/helpdesk?SGWID=0-1723113-12-971304-0 (Click on ⟶ Templates ⟶ LaTeX ⟶ monographs)
   Electronic material can be included if appropriate. Please contact the publisher.

3. Those considering a book which might be suitable for the series are strongly advised to contact the publisher or the series editors at an early stage.

General Remarks

Careful preparation of manuscripts will help keep production time short and ensure a satisfactory appearance of the finished book.

The following terms and conditions hold:

Regarding free copies and royalties, the standard terms for Springer mathematics textbooks hold. Please write to martin.peters@springer.com for details.

Authors are entitled to purchase further copies of their book and other Springer books for their personal use, at a discount of 33.3% directly from Springer-Verlag.

Series Editors

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
barth@nas.nasa.gov

Michael Griebel
Institut für Numerische Simulation
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
griebel@ins.uni-bonn.de

David E. Keyes
Mathematical and Computer Sciences
and Engineering
King Abdullah University of Science
and Technology
P.O. Box 55455
Jeddah 21534, Saudi Arabia
david.keyes@kaust.edu.sa

and

Department of Applied Physics
and Applied Mathematics
Columbia University
500 W. 120th Street
New York, NY 10027, USA
kd2112@columbia.edu

Risto M. Nieminen
Department of Applied Physics
Aalto University School of Science
and Technology
00076 Aalto, Finland
risto.nieminen@tkk.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
dirk.roose@cs.kuleuven.be

Tamar Schlick
Department of Chemistry
and Courant Institute
of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012, USA
schlick@nyu.edu

Editor for Computational Science
and Engineering at Springer:
Martin Peters
Springer-Verlag
Mathematics Editorial IV
Tiergartenstrasse 17
69121 Heidelberg, Germany
martin.peters@springer.com

Texts in Computational Science and Engineering

1. H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming, 2nd Edition.
2. A. Quarteroni, F. Saleri, P. Gervasio, *Scientific Computing with MATLAB and Octave*, 4th Edition.
3. H. P. Langtangen, *Python Scripting for Computational Science*, 3rd Edition.
4. H. Gardner, G. Manduchi, *Design Patterns for e-Science*.
5. M. Griebel, S. Knapek, G. Zumbusch, *Numerical Simulation in Molecular Dynamics*.
6. H. P. Langtangen, *A Primer on Scientific Programming with Python*, 4th Edition.
7. A. Tveito, H. P. Langtangen, B. F. Nielsen, X. Cai, *Elements of Scientific Computing*.
8. B. Gustafsson, *Fundamentals of Scientific Computing*.
9. M. Bader, *Space-Filling Curves*.
10. M.G. Larson, F. Bengzon, *The Finite Element Method: Theory, Implementation, and Practice*.
11. W. Gander, M.J. Gander, F. Kwok, *Scientific Computing*. An Introduction using Maple and MATLAB.
12. P. Deuflhard, S. Röblitz, *A Guide to Numerical Modelling in Systems Biology*.

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/5151

Monographs in Computational Science and Engineering

1. J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.-A. Mardal, A. Tveito, *Computing the Electrical Activity in the Heart.*

For further information on this book, please have a look at our mathematics catalogue at the following URL: www.springer.com/series/7417

Lecture Notes in Computational Science and Engineering

1. D. Funaro, *Spectral Elements for Transport-Dominated Equations.*
2. H.P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.
3. W. Hackbusch, G. Wittum (eds.), *Multigrid Methods V.*

4. P. Deuflhard, J. Hermans, B. Leimkuhler, A.E. Mark, S. Reich, R.D. Skeel (eds.), *Computational Molecular Dynamics: Challenges, Methods, Ideas.*
5. D. Kröner, M. Ohlberger, C. Rohde (eds.), *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws.*
6. S. Turek, *Efficient Solvers for Incompressible Flow Problems.* An Algorithmic and Computational Approach.
7. R. von Schwerin, ***M**ulti **B**ody **S**ystem **SIM**ulation.* Numerical Methods, Algorithms, and Software.
8. H.-J. Bungartz, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
9. T.J. Barth, H. Deconinck (eds.), *High-Order Methods for Computational Physics.*
10. H.P. Langtangen, A.M. Bruaset, E. Quak (eds.), *Advances in Software Tools for Scientific Computing.*
11. B. Cockburn, G.E. Karniadakis, C.-W. Shu (eds.), *Discontinuous Galerkin Methods.* Theory, Computation and Applications.
12. U. van Rienen, *Numerical Methods in Computational Electrodynamics.* Linear Systems in Practical Applications.
13. B. Engquist, L. Johnsson, M. Hammill, F. Short (eds.), *Simulation and Visualization on the Grid.*
14. E. Dick, K. Riemslagh, J. Vierendeels (eds.), *Multigrid Methods VI.*
15. A. Frommer, T. Lippert, B. Medeke, K. Schilling (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics.*
16. J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems.* Theory, Algorithm, and Applications.
17. B.I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition.*
18. U. van Rienen, M. Günther, D. Hecht (eds.), *Scientific Computing in Electrical Engineering.*
19. I. Babuška, P.G. Ciarlet, T. Miyoshi (eds.), *Mathematical Modeling and Numerical Simulation in Continuum Mechanics.*
20. T.J. Barth, T. Chan, R. Haimes (eds.), *Multiscale and Multiresolution Methods.* Theory and Applications.
21. M. Breuer, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
22. K. Urban, *Wavelets in Numerical Simulation.* Problem Adapted Construction and Applications.
23. L.F. Pavarino, A. Toselli (eds.), *Recent Developments in Domain Decomposition Methods.*
24. T. Schlick, H.H. Gan (eds.), *Computational Methods for Macromolecules: Challenges and Applications.*

25. T.J. Barth, H. Deconinck (eds.), *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*.
26. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations*.
27. S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*.
28. C. Carstensen, S. Funken, W. Hackbusch, R.H.W. Hoppe, P. Monk (eds.), *Computational Electromagnetics*.
29. M.A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*.
30. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders (eds.), *Large-Scale PDE-Constrained Optimization*.
31. M. Ainsworth, P. Davies, D. Duncan, P. Martin, B. Rynne (eds.), *Topics in Computational Wave Propagation*. Direct and Inverse Problems.
32. H. Emmerich, B. Nestler, M. Schreckenberg (eds.), *Interface and Transport Dynamics.* Computa- tional Modelling.
33. H.P. Langtangen, A. Tveito (eds.), *Advanced Topics in Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.
34. V. John, *Large Eddy Simulation of Turbulent Incompressible Flows.* Analytical and Numerical Results for a Class of LES Models.
35. E. Bänsch (ed.), *Challenges in Scientific Computing - CISC 2002.*
36. B.N. Khoromskij, G. Wittum, *Numerical Solution of Elliptic Differential Equations by Reduction to the Interface.*
37. A. Iske, *Multiresolution Methods in Scattered Data Modelling.*
38. S.-I. Niculescu, K. Gu (eds.), *Advances in Time-Delay Systems.*
39. S. Attinger, P. Koumoutsakos (eds.), *Multiscale Modelling and Simulation.*
40. R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Wildlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering.*
41. T. Plewa, T. Linde, V.G. Weirs (eds.), *Adaptive Mesh Refinement – Theory and Applications.*
42. A. Schmidt, K.G. Siebert, *Design of Adaptive Finite Element Software.* The Finite Element Toolbox ALBERTA.
43. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations II.*
44. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Methods in Science and Engineering.*
45. P. Benner, V. Mehrmann, D.C. Sorensen (eds.), *Dimension Reduction of Large-Scale Systems.*
46. D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems.*

47. A. Boriçi, A. Frommer, B. Joó, A. Kennedy, B. Pendleton (eds.), *QCD and Numerical Analysis III*.
48. F. Graziani (ed.), *Computational Methods in Transport.*
49. B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, A. Mark, T. Schlick, C. Schütte, R. Skeel (eds.), *New Algorithms for Macromolecular Simulation.*
50. M. Bücker, G. Corliss, P. Hovland, U. Naumann, B. Norris (eds.), *Automatic Differentiation: Applications, Theory, and Implementations.*
51. A.M. Bruaset, A. Tveito (eds.), *Numerical Solution of Partial Differential Equations on Parallel Computers.*
52. K.H. Hoffmann, A. Meyer (eds.), *Parallel Algorithms and Cluster Computing.*
53. H.-J. Bungartz, M. Schäfer (eds.), *Fluid-Structure Interaction.*
54. J. Behrens, *Adaptive Atmospheric Modeling.*
55. O. Widlund, D. Keyes (eds.), *Domain Decomposition Methods in Science and Engineering XVI.*
56. S. Kassinos, C. Langer, G. Iaccarino, P. Moin (eds.), *Complex Effects in Large Eddy Simulations.*
57. M. Griebel, M.A Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations III.*
58. A.N. Gorban, B. Kégl, D.C. Wunsch, A. Zinovyev (eds.), *Principal Manifolds for Data Visualization and Dimension Reduction.*
59. H. Ammari (ed.), *Modeling and Computations in Electromagnetics: A Volume Dedicated to Jean-Claude Nédélec.*
60. U. Langer, M. Discacciati, D. Keyes, O. Widlund, W. Zulehner (eds.), *Domain Decomposition Methods in Science and Engineering XVII.*
61. T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations.*
62. F. Graziani (ed.), *Computational Methods in Transport: Verification and Validation.*
63. M. Bebendorf, *Hierarchical Matrices.* A Means to Efficiently Solve Elliptic Boundary Value Problems.
64. C.H. Bischof, H.M. Bücker, P. Hovland, U. Naumann, J. Utke (eds.), *Advances in Automatic Differentiation.*
65. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations IV*.
66. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Modeling and Simulation in Science.*
67. I.H. Tuncer, Ü. Gülcat, D.R. Emerson, K. Matsuno (eds.), *Parallel Computational Fluid Dynamics 2007.*
68. S. Yip, T. Diaz de la Rubia (eds.), *Scientific Modeling and Simulations*.

69. A. Hegarty, N. Kopteva, E. O'Riordan, M. Stynes (eds.), *BAIL* 2008 – *Boundary and Interior Layers.*
70. M. Bercovier, M.J. Gander, R. Kornhuber, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XVIII.*
71. B. Koren, C. Vuik (eds.), *Advanced Computational Methods in Science and Engineering.*
72. M. Peters (ed.), *Computational Fluid Dynamics for Sport Simulation.*
73. H.-J. Bungartz, M. Mehl, M. Schäfer (eds.), *Fluid Structure Interaction II – Modelling, Simulation, Optimization.*
74. D. Tromeur-Dervout, G. Brenner, D.R. Emerson, J. Erhel (eds.), *Parallel Computational Fluid Dynamics 2008.*
75. A.N. Gorban, D. Roose (eds.), *Coping with Complexity: Model Reduction and Data Analysis.*
76. J.S. Hesthaven, E.M. Rønquist (eds.), *Spectral and High Order Methods for Partial Differential Equations.*
77. M. Holtz, *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance.*
78. Y. Huang, R. Kornhuber, O. Widlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering XIX.*
79. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations V.*
80. P.H. Lauritzen, C. Jablonowski, M.A. Taylor, R.D. Nair (eds.), *Numerical Techniques for Global Atmospheric Models.*
81. C. Clavero, J.L. Gracia, F. Lisbona (eds.), *BAIL 2010 – Boundary and Interior Layers, Computational and Asymptotic Methods.*
82. B. Engquist, O. Runborg, Y.R. Tsai (eds.), *Numerical Analysis and Multiscale Computations.*
83. I.G. Graham, T.Y. Hou, O. Lakkis, R. Scheichl (eds.), *Numerical Analysis of Multiscale Problems.*
84. A. Logg, K.-A. Mardal, G. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method.*
85. J. Blowey, M. Jensen (eds.), *Frontiers in Numerical Analysis - Durham 2010.*
86. O. Kolditz, U.-J. Gorke, H. Shao, W. Wang (eds.), *Thermo-Hydro-Mechanical-Chemical Processes in Fractured Porous Media - Benchmarks and Examples.*
87. S. Forth, P. Hovland, E. Phipps, J. Utke, A. Walther (eds.), it Recent Advances in Algorithmic Differentiation.
88. J. Garcke, M. Griebel (eds.), *Sparse Grids and Applications.*
89. M. Griebel, M. A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations VI.*

90. C. Pechstein, *Finite and Boundary Element Tearing and Interconnecting Solvers for Multiscale Problems.*
91. R. Bank, M. Holst, O. Widlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering XX*.
92. H. Bijl, D. Lucor, S. Mishra, C. Schwab (eds.), *Uncertainty Quantification in Computational Fluid Dynamics*.
93. M. Bader, H.-J. Bungartz, T. Weinzierl (eds.), *Advanced Computing*.
94. M. Ehrhardt, T. Koprucki (eds.), *Advanced Mathematical Models and Numerical Techniques for Multi-Band Effective Mass Approximations*.
95. M. Azaïez, H. El Fekih, J.S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2012*.
96. F. Graziani, M.P. Desjarlais, R. Redmer, S.B. Trickey (eds.), *Frontiers and Challenges in Warm Dense Matter*.
97. J. Garcke, D. Pflüger (eds.), *Sparse Grids and Applications – Munich 2012*.
98. J. Erhel, M. Gander, L. Halpern, G. Pichot, T. Sassi, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XXI*.
99. R. Abgrall, H. Beaugendre, P.M. Congedo, C. Dobrzynski, V. Perrier, M. Ricchiuto (eds.), *High Order Nonlinear Numerical Methods for Evolutionary PDEs - HONOM 2013*.
100. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations VII*.
101. R. Hoppe (ed.), *Optimization with PDE Constraints – ESF Networking Program 'OPTPDE'*.
102. S. Dahlke, W. Dahmen, M. Griebel, W. Hackbusch, K. Ritter, R. Schneider, C. Schwab, H. Yserentant (eds.), *Extraction of Quantifiable Information from Complex Systems*.
103. A. Abdulle, S. Deparis, D. Kressner, F. Nobile, M. Picasso (eds.), *Numerical Mathematics and Advanced Applications - ENUMATH 2013*.
104. T. Dickopf, M.J. Gander, L. Halpern, R. Krause, L.F. Pavarino (eds.), *Domain Decomposition Methods in Science and Engineering XXII*.

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/3527