

Data Fitting and Experimental
Design in Dynamical Systems
with
EASY-FIT *Model Design*

- User's Guide -

Prof. K. Schittkowski
Department of Computer Science
University of Bayreuth
Version 5.1, 2009
Copyright ©1997-2009, Klaus Schittkowski

June 24, 2009

EASY-FIT^{*ModelDesign*} is an interactive software system to identify parameters in explicit model functions, dynamical systems of equations, Laplace transformations, systems of ordinary differential equations, differential algebraic equations, or systems of one-dimensional time-dependent partial differential equations with or without algebraic equations. Proceeding from given experimental data, i.e., observation times and measurements, the minimum least squares distance of measured data from a fitting criterion is computed, that depends on the solution of the dynamical system.

Moreover, it is possible to predetermine an optimal experimental design by fixing the model parameters. Additional design parameters, for example initial concentrations or input feeds, are used to minimize the size of confidence intervals. Weight optimization helps to identify relevant time values where experiments can be taken.

The mathematical background of the numerical algorithms is described in Schittkowski [438] in form of a comprehensive textbook. Also, the outcome of numerical comparative performance evaluations is found there, together with a chapter about numerical pitfalls, testing the validity of models, and a collection of 12 real-life case studies. Most of the case studies possess an industrial background.

The software system is implemented in form of a database under Microsoft Office Access 2007 running under Windows XP or higher, and comes with the royalty-free runtime version. The underlying numerical algorithms are coded in Fortran and are executable independently from the interface. Model functions are either interpreted and evaluated symbolically by a program called PCOMP permitting automatic differentiation of nonlinear model functions, or by user-provided Fortran subroutines. In the latter case, interfaces for the Fortran compilers Watcom F77/386, Salford FTN77, Lahey F77L-EM/32, Compaq Visual Fortran, Absoft Pro Fortran, Microsoft Fortran PowerStation, and Intel Visual Fortran for Windows 32 and Windows 64 environments are provided.

Important Notes:

1. Trademarks:

Windows, Microsoft, PowerStation are registered trademarks of Microsoft Corp.

WATCOM is a registered trademark of WATCOM Systems Inc.

FTN77 is a trademark of Salford Software Ltd.

INTEL is a trademark of Intel Corporation

Adobe, Acrobat are registered trademarks of Adobe Systems Inc.

2. Copyrights:

GNUPLLOT Copyright ©1986-1993,1998,2004, Thomas Williams, Colin Kelley

RADAU5, DOPRI5 Copyright ©2004, Ernst Hairer

LAPACK Copyright ©1992-2007, The University of Tennessee

3. Data Fitting Codes:

Note that with Version 4.32 of **EASY-FIT**^{*ModelDesign*} the data fitting algorithm DFNLP has been replaced by the codes

NLPLSQ - least squares data fitting

NLPLSX - least squares data fitting for very many measurements

NLPL1 - L_1 data fitting (sum of absolute residual values)

NLPINF - L_∞ data fitting (maximum of absolute residual values)

Disclaimer:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

0.1	Installation	i
0.1.1	Hardware and Software Requirements	i
0.1.2	Packing List	i
0.1.3	System Setup	iii
0.1.4	Starting EASY-FIT ^{<i>ModelDesign</i>}	v
0.1.5	Dimensioning Parameters	vi
1	Introduction	1
2	Data Fitting Models	1
2.1	Introduction	2
2.2	Explicit Model Functions	4
2.3	Laplace Transforms	9
2.4	Steady State Equations	11
2.5	Ordinary Differential Equations	14
2.5.1	Standard Formulation	14
2.5.2	Differential Algebraic Equations	15
2.5.3	Switching Points	17
2.5.4	Constraints	26
2.5.5	Shooting Method	28
2.5.6	Boundary Value Problems	35
2.5.7	Variable Initial Times	37
2.6	Partial Differential Equations	43
2.6.1	Standard Formulation	43
2.6.2	Partial Differential Algebraic Equations	45
2.6.3	Flux Functions	48
2.6.4	Coupled Ordinary Differential Algebraic Equations	50
2.6.5	Integration Areas and Transition Conditions	56
2.6.6	Switching Points	61
2.6.7	Constraints	65
2.7	Optimal Control Problems	71

3	Statistical Analysis and Experimental Design	79
3.1	Confidence Intervals	81
3.2	Significance Levels	83
3.3	Experimental Design	89
3.4	Experimental Design with Weights	101
4	Numerical Algorithms	1
4.1	Data Fitting Algorithms	1
4.2	Steady State Systems	2
4.3	Laplace Back-Transformation	3
4.4	Ordinary Differential Equations	4
4.5	Differential Algebraic Equations	5
4.6	Partial Differential Equations	6
4.7	Partial Differential Algebraic Equations	9
4.8	Statistical Analysis	10
5	The Modeling Language PCOMP	1
5.1	Automatic Differentiation	1
5.2	Input Format for PCOMP	4
5.3	Error Messages of PCOMP	15
6	Model Functions and Equations	1
6.1	Explicit Model Functions	1
6.2	Laplace Transformations	3
6.3	Systems of Steady State Equations	4
6.4	Ordinary Differential Equations	6
6.5	Differential Algebraic Equations	8
6.6	Time-Dependent Partial Differential Equations	10
6.7	Partial Differential Algebraic Equations	13
7	Problem Data and Solution Tolerances	1
7.1	Model Independent Information	1
7.1.1	Problem Name	1
7.1.2	Documentation Text	2
7.1.3	Parameters to be Estimated	3
7.1.4	Input Type of Model Functions	3
7.1.5	Numerical Analysis	4
7.1.6	Optimization Tolerances	6
7.1.7	Scaling	11
7.1.8	Number of Plot Points	11
7.1.9	Logarithmic Plot	11
7.1.10	Experimental Data	11

7.1.11	Data Fitting Norm	12
7.2	Model Dependent Information	13
7.2.1	Model Data for Explicit Functions	13
7.2.2	Model Data for Steady State Equations	16
7.2.3	Model Data for Laplace Transformations	20
7.2.4	Model Data for Ordinary Differential Equations	21
7.2.5	Model Data for Differential Algebraic Equations	26
7.2.6	Model Data for Time-Dependent Partial Differential Equations	33
7.2.7	Model Data for PDAE's	40
8	Menu Commands	1
8.1	File Command	1
8.2	Edit Command	5
8.3	Start Command	6
8.4	Report Command	7
8.5	Data Command	10
8.6	Delete Command	14
8.7	Make Command	15
8.8	Utilities Command	16
9	External Usage of Numerical Codes	1
9.1	MODFIT: Explicit Model Functions, Dynamical Systems, ODE's and DAE's	1
9.2	PDEFIT: Partial Differential Equations	28
10	Test Examples	1
10.1	Explicit Model Functions	3
10.2	Laplace Transforms	14
10.3	Steady State Equations	15
10.4	Ordinary Differential Equations	18
10.5	Differential Algebraic Equations	41
10.6	Partial Differential Equations	45
10.7	Partial Differential Algebraic Equations	60

0.1 Installation

EASY-FIT^{*ModelDesign*} consists of a database containing models, data and results, and of underlying numerical algorithms for solving the parameter estimation problem depending on the mathematical structure, i.e.

- MODFIT parameter estimation in explicit functions, steady state equations, Laplace transforms, ordinary differential and differential algebraic equations
- PDEFIT parameter estimation in one-dimensional time-dependent partial differential equations and partial differential algebraic equations

By the following notes, the system installation and hardware requirements are outlined.

0.1.1 Hardware and Software Requirements

Installation of **EASY-FIT**^{*ModelDesign*} requires 95 MB on hard disk plus 130 MP for the Microsoft Office Access 2007 runtime version. The program runs under Windows XP or higher. **EASY-FIT**^{*ModelDesign*} comes with the royalty-free runtime version of Microsoft Office Access 2007 (English).

All model functions are defined in the PCOMP modelling language to be interpreted and evaluated during run time. Derivatives, as far as needed, are computed by automatic differentiation. The full version of **EASY-FIT**^{*ModelDesign*} allows also the most flexible input of the underlying model functions in form of Fortran code, and has interfaces for Compaq Visual Fortran, Watcom F77/386, Salford FTN77, Lahey F77L-EM/32, Absoft Pro Fortran, Microsoft Fortran PowerStation, and Intel Visual Fortran for Windows 32 and Windows 64 environments, where the compiler and linker options can be altered and adapted interactively.

0.1.2 Packing List

Basically, **EASY-FIT**^{*ModelDesign*} consists of a user interface in form of a database implemented in Microsoft Office Access 2007, and some numerical routines. The following essential files and directories are submitted:

Numerical codes:

MODFIT.EXE	Solving parameter estimation problems in explicit models, time-dependent algebraic equations, ordinary differential equations, differential algebraic systems, and Laplace transforms
MODFIT.FOR	Corresponding Fortran source code (only complete version)
MODFUN_E.FOR	Frame of a Fortran code for estimating parameters in explicit model functions
MODFUN_O.FOR	Frame of a Fortran code for estimating parameters in differential equations
MODFUN_A.FOR	Frame of a Fortran code for estimating parameters in differential algebraic equations
MODFUN_S.FOR	Frame of a Fortran code for estimating parameters in steady state systems
MODFIT.INC	Include file with dimensioning parameters for MODFIT
PDEFIT.EXE	Solving parameter estimation problems in systems of one-dimensional partial differential equations and partial differential algebraic equations
PDEFIT.FOR	Corresponding Fortran source code (only complete version)
PDEFUN.FOR	Frame of a Fortran code for estimating parameters in systems of partial differential equations
PDEFIT.INC	Include file with dimensioning parameters for PDEFIT
COMPILE.BAT	DOS batch file to execute the Fortran compiler, can be modified interactively
LINKER.BAT	DOS batch file to execute the Fortran linker, can be modified interactively

Plot programs and editor:

SP_PLOT.EXE	Standard plot program, where input data are read from files
GNUPLOT.EXE	Public domain plot program Gnuplot
EDITOR.EXE	Syntax-highlighting external editor

Database:

EASY_FIT.MDE	Main database of EASY-FIT <i>ModelDesign</i> containing tables, forms, reports, macros and modules
EASY_FIT.HLP	Corresponding help file
EASY_FIT.ICO	Icon file for EASY-FIT <i>ModelDesign</i>
EASY_FIT.PDF	Adobe Acrobat Reader file containing complete documentation

Subdirectories:

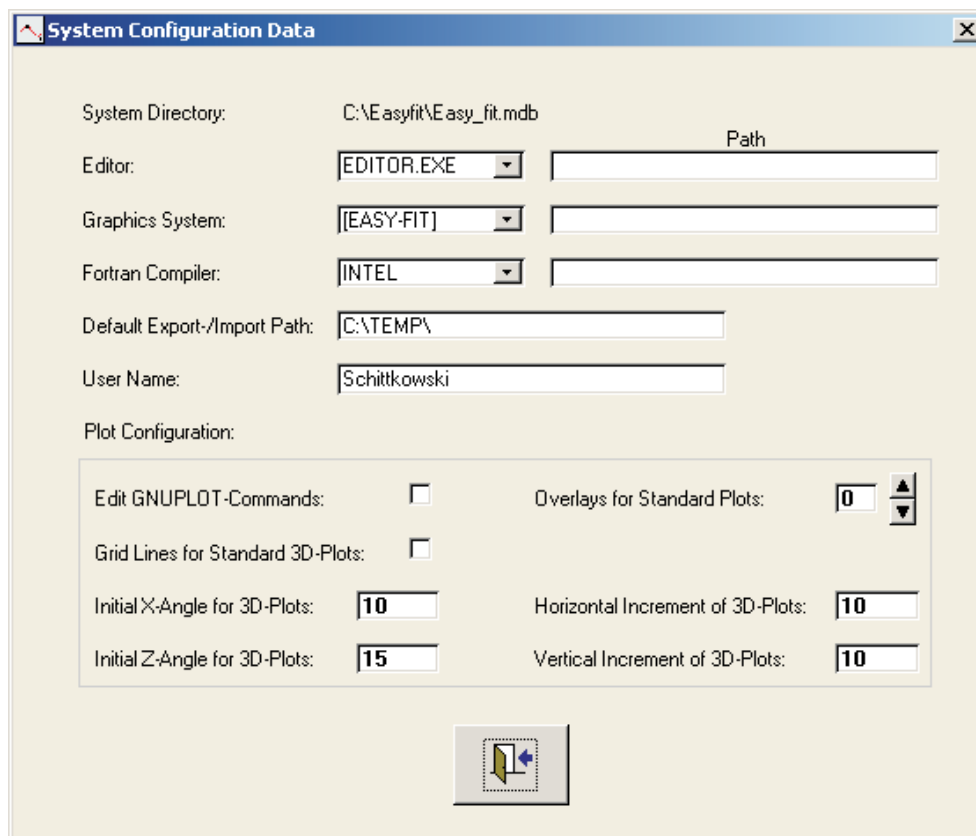
OBJECT	Directory with object codes for the Watcom, Salford, Lahey, Compaq, Absoft, Microsoft, and Intel Fortran compilers containing underlying optimization algorithms and ODE/PDE-solvers (only complete version)
PROBLEMS	Directory for test example files with extensions < * >.FUN and < * >.FOR

0.1.3 System Setup

Download the file EASYFIT.EXE and start the installation by clicking on this file. In case of a local network, administrator rights are required. If there exists an older version of **EASY-FIT***ModelDesign*, it is recommended to save first all problems of interest to a temporary directory. After successful installation, the saved problems can be imported again. **EASY-FIT***ModelDesign* comes with the royalty-free runtime version of Microsoft Office Access 2007.

When starting **EASY-FIT***ModelDesign* the first time after a successful setup, a couple of directory strings are inserted automatically into an internal table. They can be adapted to a special situation depending on the environment given. Alterations can be made by the Utilities command in the menu bar. E.g., the favorite text editor may be defined to be used for input and modification of model functions.

In more detail, the following configuration information is available:



System Configuration Data

System Directory: C:\Easyfit\Easy_fit.mdb

Editor: EDITOR.EXE Path

Graphics System: [EASY-FIT]

Fortran Compiler: INTEL

Default Export-/Import Path: C:\TEMP\

User Name: Schittkowski

Plot Configuration:

Edit GNUPLOT-Commands: <input type="checkbox"/>	Overlays for Standard Plots: 0
Grid Lines for Standard 3D-Plots: <input type="checkbox"/>	
Initial X-Angle for 3D-Plots: 10	Horizontal Increment of 3D-Plots: 10
Initial Z-Angle for 3D-Plots: 15	Vertical Increment of 3D-Plots: 10




Figure 1: Configuration Form

Name	Default	Contents
system directory	C:\EASYFIT\	EASY-FIT <i>ModelDesign</i> main directory with database, help files etc.
editor	[EASY-FIT]	Internal or alternative external editor, for example EDITOR.EXE
graphics system	[EASY-FIT]	Internal or alternative external graphics system identified by string GNUPLOT
Fortran compiler	SALFORD	Available Fortran compiler. Insert ABSOFT, WATCOM, SALFORD, LAHEY, V_FORTTRAN, MS_POWER, INTEL, or INTEL64 for Absoft Pro Fortran, Watcom F77/386, Salford FTN77, Lahey F77L-EM/32, Compaq Visual Fortran, Microsoft Fortran PowerStation, Intel Visual Fortran compiler (IA32) and Intel Visual Fortran 64 bit compiler (EM64T), respectively

In case of installing an **EASY-FIT***ModelDesign* version coming with all object files, proceed as follows. First, the submitted object codes of the numerical algorithms must be copied to a subdirectory with name OBJECT. The object codes of the driving routines with names MODFIT*.FOR and PDEFIT*.FOR can be generated subsequently, if necessary. The easiest way is to edit dimensioning parameters in the Utilities menu and to let these modules be compiled automatically by **EASY-FIT***ModelDesign*. If the submitted default compiler interface is to be changed, e.g., from the Watcom to the Salford or Lahey compiler, the user has to set corresponding compiler name, some path names, and the compiler and linker execution commands.

An external editor can be used to create or modify model functions either in the PCOMP or the Fortran language. Note that **EASY-FIT***ModelDesign* is delivered with two editors, an internal GUI form ([EASY-FIT]) and an external executable one with syntax highlighting (EDITOR.EXE). Both allow direct parse of PCOMP code or compilation and link of Fortran code. To use the external editor, the file EDITOR.EXE must be part of the **EASY-FIT***ModelDesign* installation directory.

0.1.4 Starting **EASY-FIT***ModelDesign*

It is recommended to start **EASY-FIT***ModelDesign* always from its shortcut in the program menu generated by the setup program, or from a corresponding desktop icon to avoid conflicts with an existing Microsoft Office Access version. The welcome window of **EASY-FIT***ModelDesign* is displayed and the main form of the database is opened.

If the main form cannot be opened correctly, please check the language settings. Non-unicode languages like Chinese, Arabic or other settings cause some problems.

If the database reacts too slow, for example when starting a data fitting code or when displaying a report, delete a certain subset of problems you do not need.

The file README.TXT contains last-minute changes, a summary of new features and especially the information how to transfer parameter estimation problems from easier versions of **EASY-FIT**^{*ModelDesign*} to the new one.

Welcome Form

EASY-FIT^{*ModelDesign*}

is an interactive software system to identify parameters in systems of

- ... explicit model functions
- ... dynamical equations (steady-state systems)
- ... Laplace equations
- ... ordinary differential equations (stiff/non-stiff)
- ... differential algebraic equations
- ... one-dimensional, time-dependent partial differential equations
- ... one-dimensional partial differential algebraic equations

Copyright: Prof. K. Schittkowski
Dept. of Computer Science
University of Bayreuth
D - 95440 Bayreuth

Synonyms: Parameter estimation, data fitting, experimental design,
mathematical modelling, simulation

Important:

1. Usage is restricted according to the valid license agreement.
2. Select favorite editor and graphics system in subsequent form or by 'Utility' menu item 'System Configuration'.
3. Check the compiler and linker calling sequences before generating own Fortran code.
4. You are NOT allowed to distribute this software to any other person or organization.
5. If the system reacts too slow on your computer, please delete subset of demo problems you do not need.

Figure 2: Welcome Form

If something goes wrong, please contact the author under

phone : +49 (0)921 553278
 fax : +49 (0)921 35557
 e-mail : klaus.schittkowski@uni-bayreuth.de
 home page : <http://www.klaus-schittkowski.de>

0.1.5 Dimensioning Parameters

The numerical algorithms require dimensioning parameters for defining working arrays of suitable lengths. They serve also as upper bounds for certain model parameters, i.e., maximum number of variables to be estimated or maximum number of measurements. Whereas the full version can be adapted to any size, there are some restrictions for the demo version:

Number of parameters to be estimated:	2
Number of equations:	2
Number of constraints:	5
Number of ODEs of discretized PDE:	100
Number of time values:	20
Number of measurement sets:	2
Number of measurements:	40

To find out the allowed maximum problem sizes of the full version, one should investigate the corresponding include files MODFIT.INC and PDEFIT.INC from the utilities command of the menu bar. New executable files can be linked subsequently, if any of the bounds are changed, and if object codes are available. The meaning of the parameters used is completely described by initial comments.

Chapter 1

Introduction

Parameter estimation plays an important role in natural science, engineering, and many other disciplines. The key idea is to estimate unknown parameters p_1, \dots, p_n of a mathematical model that describes a real life situation, by minimizing the distance of some known experimental data from theoretically predicted values of a model function at certain time values. Thus, also model parameters that cannot be measured directly, can be identified by a least squares fit and analyzed subsequently in a quantitative way.

In mathematical and somewhat simplified notation, we want to solve a least squares problem of the form

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^l (h(p, y(p, t_i), t_i) - y_i)^2 \\ & p_l \leq p \leq p_u \quad , \end{aligned} \quad (1.1)$$

where $h(p, y, t)$ is a fitting function depending on the unknown parameter vector p , the time t , and the solution $y(p, t)$ of an underlying dynamical system. A typical dynamical system is given by differential equations that describe a time-dependent process, and that depend on the parameter vector p . Instead of minimizing the sum of squares, we may apply alternative residual norms, for example with the goal to minimize the sum of absolute residual values or the maximum of absolute residual values.

Parameter estimation, also called parameter identification, nonlinear regression, or data fitting, is extremely important in all practical situations, where a mathematical model and corresponding experimental data are available to analyze the behavior of a dynamical system.

The main goal of the documentation is to introduce some numerical methods that can be used to compute parameters by a least squares fit in form of a *toolbox*. The mathematical model that is set up by a system analyst, has to belong to one of the following categories,

- explicit model functions,
- steady state systems,
- Laplace transforms of differential equations,

- ordinary differential equations,
- differential algebraic equations,
- one-dimensional time-dependent partial differential equations,
- one-dimensional partial differential algebraic equations.

To understand at least some of the basic features of the presented algorithms, to apply available software, and to analyze numerical results, it is necessary to combine knowledge from many different mathematical disciplines, for example

- modelling,
- nonlinear optimization,
- system identification,
- numerical solution of ordinary differential equations,
- discretization of partial differential equations,
- sensitivity analysis,
- automatic differentiation,
- Laplace transforms,
- statistics.

The mathematical background of the numerical algorithms is described in Schittkowski [438] in form of a comprehensive textbook. Also, the outcome of numerical comparative performance evaluations is found there, together with a chapter about numerical pitfalls, testing the validity of models, and a collection of 12 real-life case studies. Most of the case studies possess an industrial background.

The general mathematical model to be investigated contains certain features to apply the numerical methods to a large set of practically relevant situations. Some of the most important issues are:

1. More than one fitting criterion can be defined, i.e., more than one experimental data set can be fitted within a model formulation.
2. The fitting criteria are arbitrary functions depending on the parameters to be estimated, the solution of the underlying dynamical system, and the time variable.
3. The model may possess arbitrary equality or inequality constraints with respect to the parameters to be estimated, and upper and lower bounds for the parameters.

4. Model equations may contain an additional independent parameter, for example experimental concentration or temperature values.
5. Differential-algebraic equations can be solved up to index 3. Consistent initial values for index-1-formulations are computed internally.
6. In case of partial differential equations, also coupled ordinary differential equations and non-continuous transitions for state variable and flux between different areas can be taken into account.
7. Differential equation models may possess additional break or switching points, where the model dynamics is changed and where integration is restarted, for example if a new dose is applied in case of a pharmacokinetic model.
8. The switching points mentioned before, may become optimization variables to allow the modeling of dynamical input, for instance, to compute optimal bang-bang feed controls of a chemical reactor.
9. The model functions may be defined by their Laplace transforms, where the back-transformation is performed numerically.
10. Gradients can be evaluated by automatic differentiation without additional round-off, truncation or approximation errors, and without compiling and linking of code.
11. Ordinary differential equations may become stiff and large. We introduce explicit and implicit methods and exploit band structures.
12. Parameter estimation problems based on unstable differential equations can be solved by the shooting method.
13. Various types of one-dimensional partial differential equations are permitted, also hyperbolic ones describing shock waves. Advection, diffusion, transport, or related equations can be solved successfully by non-oscillatory discretization schemes, even with non-continuous initial or boundary conditions.
14. Partial differential equations may be defined with Neumann and Dirichlet boundary or transitions conditions. Moreover, these conditions can be formulated in terms of algebraic equations coupled at arbitrary spatial positions.
15. Algebraic partial differential equations may be added to the time-dependent ones.
16. Data can be fitted with respect to the L_2 -, the L_1 -, or the L_∞ -norm, i.e., with respect to sum of squares, sum of absolute values, or maximum of absolute values of the residuals.

17. A statistical analysis provides confidence intervals for parameters depending on an user-provided estimate for the variance. Moreover, correlation coefficients and covariance matrix are computed.
18. Proceeding from the inverse of the Fisher information matrix, an eigenvalue/eigenvector analysis is performed to identify significant parameter levels for subsequent elimination of non-relevant parameters or further statistical design investigations.

Only for illustration purposes we denote the first independent model variable the *time* variable of the system, the second one the *concentration* variable and the dependent data as *measurement* values of an *experiment*. These words describe their probably most frequent usage in a practical situation. On the other hand, the terms may get any other meaning depending on the underlying application problem.

Due to the practical importance of parameter estimation, very many numerical codes have been developed in the past and are distributed within software packages. However, there is no guarantee that a mathematical algorithm is capable to solve the problem we are interested in. Possible traps preventing a solution in the desired way, are

- approximation of a local solution that is unacceptable,
- round-off errors because of an inaccurate iterative solution of the dynamical system,
- narrow curved valleys where progress towards the solution is hard to achieve,
- very flat objective function in the neighborhood of a solution, for example, when there are large perturbations in measurement data,
- overdetermined models in case of too many model parameters to be estimated, leading to infinitely many solution vectors,
- bad starting values for parameters requiring a large number of steps,
- badly scaled model functions and, in particular, measurement values,
- non-differentiable model functions.

We have to know that all efficient optimization algorithms developed for the problem class we are considering, require differentiable fitting criteria and the availability of a starting point from which the iteration cycle is initiated. Additional difficulties arise in the presence of nonlinear constraints, in particular if they are badly stated, ill-conditioned, badly scaled, linearly dependent, or, worst of all, contradictory.

Thus, users of parameter estimation software are often faced with the situation that the algorithm is unable to get a satisfactory return subject to the given solution tolerances, and that one has to restart the solution cycle by changing tolerances, internal algorithmic

decisions, or at least the starting values to get a better result. To sum up, a *black box* approach to solve a parameter estimation problem does not exist and a typical life cycle of a solution process consists of stepwise redesign of solution data.

The parameter estimation problem, alternative phrases are data fitting or system identification, is outlined in Chapter 2. It is shown, how the dynamical systems have to be adapted to fit into the least squares formulation required for starting an optimization algorithm.

A first question is always how to get suitable confidence intervals for the estimated parameters. This is one of the main investigations when analyzing the output of data fitting. Related problems are whether it is possible at all to identify parameters, or how to eliminate redundant ones, as will be discussed in the subsequent sections. Another important question is experimental design, where we want to create or improve existing experimental conditions. The goals are to reduce the number of costly experiments, to reduce error variances, or to get identifiable parameters. The corresponding tools are summarized in Chapter 3.

A brief review of the numerical algorithms implemented, is presented in Chapter 4. Only some basic features of the underlying ideas are presented. More details are found in the references and in particular in Schittkowski [438]. The codes allow the numerical identification of parameters in any of the six situations under investigation. The executable files are called MODFIT.EXE and PDEFIT.EXE.

Nonlinear model functions can be evaluated symbolically. Thus, any compilation and link of Fortran subroutines is not required whenever model functions are defined or altered in this way. A particular advantage of this approach is the automatic differentiation of model functions to avoid numerical truncation errors. The corresponding program is called PCOMP, see Dobmann, Liepelt and Schittkowski [115], and is part of the executable codes. The automatic differentiation algorithm, the PCOMP language and error messages of the parser are described in Chapter 5.

Model functions must be provided by the user either in form of the PCOMP language mentioned above, or in form of Fortran code. In the latter case, the preparation of function and gradient values is described by initial comments of a code inserted by **EASY-FIT**^{ModelDesign} as a frame. In case of PCOMP input, the order in which variables and functions are to be inserted, identifies their role in the mathematical model. A full documentation of the model function input in this situation is presented in Chapter 6.

The interactive system **EASY-FIT**^{ModelDesign} proceeds from a database for storing model information, experimental data and results. A complete context sensitive help option is included containing additional technical and organizational information about the input of data and optimization tolerances, for example. A brief outline of data organization and input is found in Chapter 7. The corresponding menu commands to define or alter data and functions, to start an optimization run or to get reports on numerical results, are described in Chapter 8.

The numerical parameter estimation codes MODFIT and PDEFIT can be executed also outside of the interactive user interface. A possible reason could be the solution of a large number of parameter estimation problems controlled by a separate command shell. In this

case, a data input file is required that contains all information for starting the numerical algorithm. The format of this file is documented in Chapter 9 in detail. Usage of the codes is illustrated by a few test examples.

The database of the delivered **EASY-FIT**^{*ModelDesign*} version contains 1,300 academic and *real life* examples, i.e., test problems with some realistic practical background. Application areas are pharmacy, biochemistry, chemical engineering, and mechanical engineering. The purpose for attaching a comprehensive collection of test problems in Chapter 10, is to become familiar with the PCOMP language and the implementation of a new model, since a large variety of different model structures is offered. The problems can be used for selecting a reference example when trying to install own dynamical models, or to test the accuracy or efficiency of the algorithms available within **EASY-FIT**^{*ModelDesign*}.

Chapter 2

Data Fitting Models

Our goal is to estimate parameters in

- explicit model functions,
- Laplace transforms,
- steady state systems,
- systems of ordinary differential equations
- systems of differential algebraic equations,
- systems of one-dimensional, time-dependent partial differential equations,
- systems of one-dimensional partial differential algebraic equations.

Proceeding from given experimental data, i.e., observation times and measurements, the minimum least squares distance of measured data from a fitting criterion is to be computed that depends on the solution of the dynamical system.

In this chapter, we summarize in detail, how the model functions $f_i(p)$ depend on the solution of a dynamical system. Moreover, we describe a couple of extensions of the data fitting problem and the dynamical system to be able to treat also more complex practical models. Most examples contain the name of the corresponding test problem of the **EASY-FIT**^{*ModelDesign*} database, from where implementation details and further data can be retrieved, in some cases only in modified form.

2.1 Introduction

The basic mathematical model is the least squares problem to minimize a sum of squares of nonlinear functions of the form

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^l f_i(p)^2 \\ & p_l \leq p \leq p_u \end{aligned} \quad . \quad (2.1)$$

Here, we assume that the parameter vector p is n -dimensional and that all nonlinear functions are continuously differentiable with respect to p . Upper and lower bounds are included to restrict the search area. $f_i(p)$ is a suitable fitting criterion which may depend on the solution of an underlying dynamical system, e.g., a system of ordinary differential equations.

Alternatively the L_2 -norm may be changed to another one, e.g., to minimize the maximum distance of experimental data from a model function. Thus, we formulate either the L_1 -problem

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^l |f_i(p)| \\ & p_l \leq p \leq p_u \end{aligned} \quad (2.2)$$

or the L_∞ -problem

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \max_{i=1, \dots, l} |f_i(p)| \\ & p_l \leq p \leq p_u \end{aligned} \quad . \quad (2.3)$$

However, we assume that our models a dynamic, i.e., depend on an additional parameter, in most cases the time. In addition, there might be an additional independent model parameter by which, e.g., a concentration or temperature value is to be specified from where a set of measurements is obtained.

To illustrate the situation, we omit possible additional data sets, dependencies on underlying dynamical systems, and constraints on the parameters, and differ between three situations.

1. Time-dependent models: The model function $f_i(p)$ depends on the experimental time, i.e., we have measurements of the form

$$(t_i, y_i) , \quad i = 1, \dots, l , \quad (2.4)$$

moreover a model function $h(p, t)$, and we want to estimate the parameter vector p by minimizing

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^l (h(p, t_i) - y_i)^2 \\ & p_l \leq p \leq p_u \end{aligned} \quad . \quad (2.5)$$

In this case, we define

$$f_i(p) = h(p, t_i) - y_i, \quad i = 1, \dots, l . \quad (2.6)$$

2. Time- and concentration dependent fitting criteria: The data fitting function $f_i(p)$ depends on the experimental time and an additional parameter which we call *concentration*. Any other physical meaning is, of course, allowed. We proceed from measurements of the form

$$(t_i, c_i, y_i) , \ i = 1, \dots, l , \quad (2.7)$$

a model function $h(p, t, c)$, and we want to estimate the parameter vector p by minimizing

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^l (h(p, t_i, c_i) - y_i)^2 \\ & p_l \leq p \leq p_u \end{aligned} . \quad (2.8)$$

In this case, we define

$$f_i(p) = h(p, t_i, c_i) - y_i, \ i = 1, \dots, l . \quad (2.9)$$

Advantages are the possibilities to define a model as a function of t and c , and to generate three-dimensional plots. The drawback of this formulation, however, is that an underlying differential equation cannot depend on c as well, since we would have to evaluate the right-hand side of an equation also at intermediate times and would not know how to insert a suitable concentration value.

3. Time- and concentration dependent models: To overcome the drawback mentioned above, we assume that the dynamical model, say an ordinary differential equation, depends on an additional, in the statistical sense independent parameter c , i.e., c may be inserted into initial values, right-hand sides, fitting criterion, or even constraints. Now we proceed from measurements of the form

$$(t_i, c_j, y_{ij}) , \ i = 1, \dots, l_t, \ j = 1, \dots, l_c . \quad (2.10)$$

The model function is again given in the form $h(p, t, c)$, and we want to estimate the parameter vector p by minimizing

$$p \in \mathbb{R}^n : \quad \begin{aligned} & \min \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (h(p, t_i, c_j) - y_{ij})^2 \\ & p_l \leq p \leq p_u \end{aligned} . \quad (2.11)$$

Now we get the fitting criterion

$$f_{ij}(p) = h(p, t_i, c_j) - y_{ij}, \ i = 1, \dots, l_t; \ j = 1, \dots, l_c . \quad (2.12)$$

In the subsequent sections, we proceed from the most general situation (2.11) and illustrate our approaches by examples.

2.2 Explicit Model Functions

In this section, we restrict our investigations to parameter estimation problems, where one vector-valued model function is available in explicit form, the so-called *fitting criterion*, with one additional variable called *time*, and optionally with another one called *concentration*. We proceed now from r measurement sets, given in the form

$$(t_i, c_j, y_{ij}^k), \quad i = 1, \dots, l_t, \quad j = 1, \dots, l_c, \quad k = 1, \dots, r, \quad (2.13)$$

where l_t time values, l_c concentration values and $l = l_t l_c r$ corresponding measurement values are defined. Together with a vector-valued model function

$$h(p, t, c) = (h_1(p, t, c), \dots, h_r(p, t, c))^T,$$

we get a data fitting formulation (2.1), (2.2), or (2.3), by

$$f_s(p) = w_{ij}^k (h_k(p, t_i, c_j) - y_{ij}^k), \quad (2.14)$$

where s runs from 1 to $l = l_t l_c r$ in any order. Moreover, we assume that there are suitable weight factors $w_{ij}^k \geq 0$ given by the user that are to reflect the individual influence of a measurement on the whole experiment. Zero weights can be defined, if, for example, there are several concentration values c_1, \dots, c_{l_c} , but measurements are not available for each time value t_1, \dots, t_{l_t} .

The basic idea is to minimize the distance between the model function at certain time and concentration points and the corresponding measurement values. This distance is denoted as the residual of the problem. In the ideal case, the residuals are zero indicating a perfect fit of the model function by the measurements.

In addition, we allow any nonlinear restrictions on the parameters to be estimated, in form of general equality or inequality constraints

$$\begin{aligned} g_j(p) &= 0, \quad j = 1, \dots, m_e, \\ g_j(p) &\geq 0, \quad j = m_e + 1, \dots, m_r. \end{aligned} \quad (2.15)$$

It must be assumed that all constraint functions are continuously differentiable with respect to p .

To summarize, the resulting least squares problem is of the form

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, t_i, c_j) - y_{ij}^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0, \quad j = 1, \dots, m_e, \\ & g_j(p) \geq 0, \quad j = m_e + 1, \dots, m_r, \\ & p_l \leq p \leq p_u, \end{aligned} \quad (2.16)$$

see (2.1). Alternatively we get the corresponding L_1 -formulation by minimizing

$$\sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} w_{ij}^k |h_k(p, t_i, c_j) - y_{ij}^k| ,$$

see (2.2), or the L_∞ -formulation

$$\max_{k=1, \dots, r; i=1, \dots, l_t; j=1, \dots, l_c} w_{ij}^k |h_k(p, t_i, c_j) - y_{ij}^k| ,$$

see (2.3).

Example 2.1 (RAT_APP) We want to fit some parameters p_1, \dots, p_4 , so that the data of Table 2.1 are approximated by a rational function

$$h(p, t) = p_1 \frac{t^2 + p_2 t}{t^2 + p_3 t + p_4} ,$$

see Lindström [288] and Deufhard, Apostolescu [112]. There is no concentration parameter, but we want to fit the outer measurement values exactly, i.e., we define two additional non-linear equality constraints $g_1(p) = h(p, t_1) - y_1$ and $g_2(p) = h(p, t_l) - y_l$ with $l = 11$. There is only one measurement set and all weights are set to 1. Thus, the least squares data fitting problem is

$$\begin{aligned} \min \quad & \sum_{i=1}^l (h(p, t_i) - y_i)^2 \\ p \in \mathbb{R}^4 : \quad & g_1(p) = 0 \quad , \\ & g_2(p) = 0 \quad . \end{aligned}$$

When starting the code DFNLP of Schittkowski [429] from

$$p^0 = (0.25, 0.39, 0.415, 0.39)^T$$

with a termination tolerance of 10^{-12} , we get the solution vector

$$p^* = (0.1923, 0.4040, 0.2750, 0.2068)^T$$

after 10 iterations. The final residual is $3.78 \cdot 10^{-3}$ and the maximum constraint violation is $3.1 \cdot 10^{-13}$. The individual residuals and the relative errors are also listed in Table 2.1. Model function and data are plotted in Figure 2.1.

Since the model function $h(p, t, c)$ does not depend on the solution of an additional dynamical system, we call it an explicit model function. Otherwise, $h(p, t, c)$ may depend on the solution vector $y(p, t, c)$ of an auxiliary problem, for example an ordinary differential equation that is implicitly defined. Models of this kind are considered in the subsequent sections. But explicit model functions can reflect solutions of dynamical systems, that are analytically solvable, as shown by the subsequent example.

t_i	y_i	$ h(p^*, t_i) - y_i $	$error$
0.0625	0.0246	$2.5230 \cdot 10^{-14}$	0.0 %
0.0714	0.0235	$4.6890 \cdot 10^{-3}$	20.0 %
0.0823	0.0323	$2.7982 \cdot 10^{-4}$	0.9 %
0.1	0.0342	$5.4681 \cdot 10^{-3}$	16.0 %
0.125	0.0456	$3.9113 \cdot 10^{-3}$	8.6 %
0.167	0.0627	$2.6394 \cdot 10^{-3}$	4.2 %
0.25	0.0844	$8.5962 \cdot 10^{-3}$	10.2 %
0.5	0.16	$1.3765 \cdot 10^{-2}$	8.6 %
1.0	0.1735	$8.6748 \cdot 10^{-3}$	5.0 %
2.0	0.1947	$3.6381 \cdot 10^{-4}$	0.2 %
4.0	0.1957	$8.0491 \cdot 10^{-16}$	0.0 %

Table 2.1: Experimental Data and Final Residuals

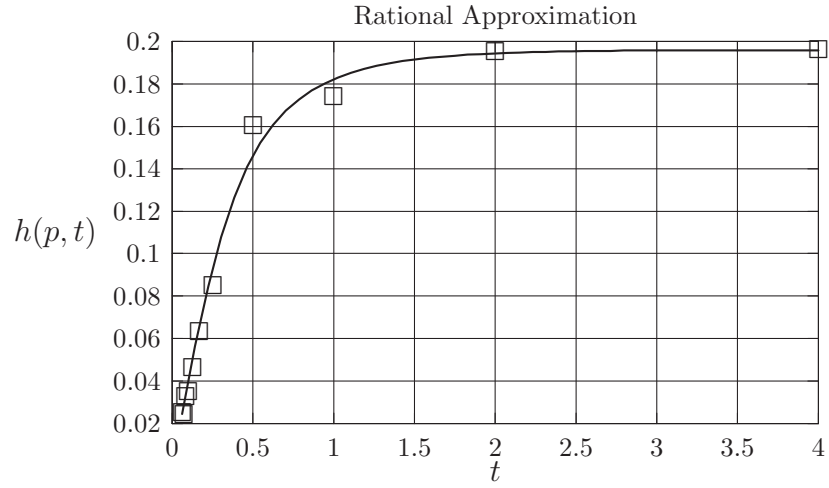


Figure 2.1: Function and Data Plot

Linear Compartmental Model

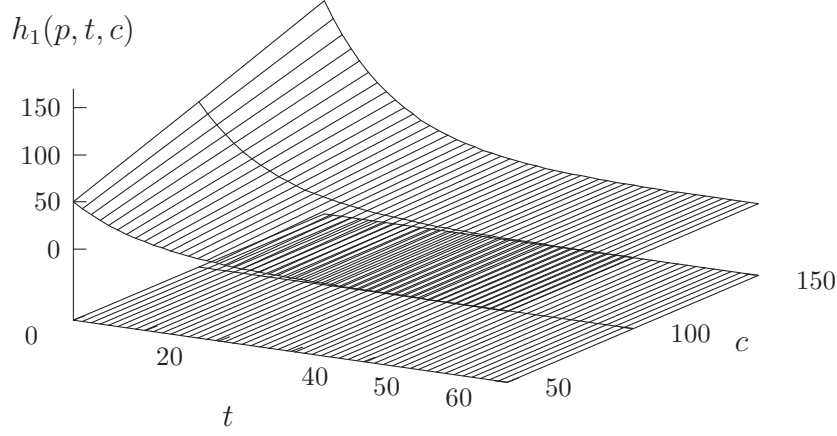


Figure 2.2: Function and Data Plot

Example 2.2 (LKIN_X3) *The next test case consists of an explicit solution of a linear ordinary differential equation*

$$\begin{aligned} h_1(p, t, c) &= c \exp(-p_1 t) , \\ h_2(p, t, c) &= \frac{p_1 c}{p_1 - p_2} (\exp(-p_2 t) - \exp(-p_1 t)) . \end{aligned}$$

The concentration parameter represents now the initial dose of an experiment, and is set to $c_1 = 50$, $c_2 = 100$, and $c_3 = 150$. We assume that there are measurements for both fitting criteria at time values 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 60. Experimental data are simulated by inserting $p_1 = 0.1$ and $p_2 = 0.05$. Subsequently a noise of 5 % is added randomly to the data. In other words we have $n = 2$, $l_t = 13$, $l_c = 3$, and $r = 2$, i.e., a set of $l = 84$ measurements. The minimization problem is

$$\min_{p \in \mathbb{R}^2} \sum_{i=1}^{13} \sum_{j=1}^3 \left(h_1(p, t_i, c_j) - y_{ij}^1 \right)^2 + \left(h_2(p, t_i, c_j) - y_{ij}^2 \right)^2 ,$$

see (2.16). Starting from $p_1^0 = 1$ and $p_2^0 = 0.1$, DFNLP computes the solution $p_1^ = 0.10014$, $p_2^* = 0.04987$ after 26 iterations. Final termination accuracy is set to 10^{-10} , and the surface plot of both fitting criteria is shown in Figures 2.2 and 2.3.*

Linear Compartmental Model

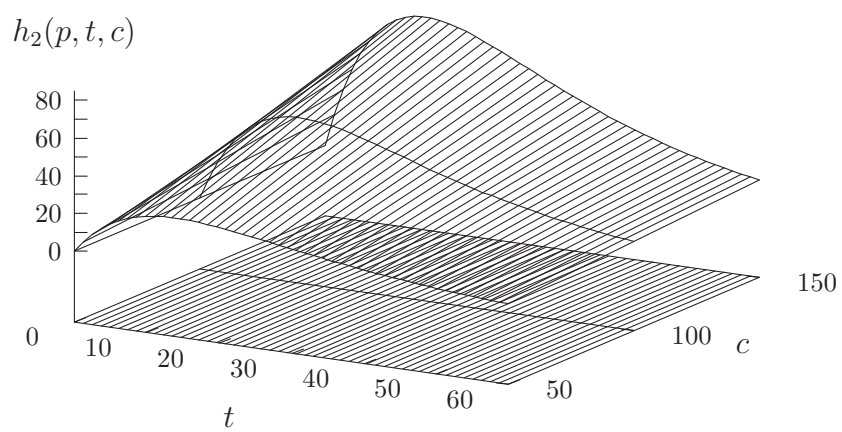


Figure 2.3: Model Function Plot

2.3 Laplace Transforms

In many practical applications, the model is available in form of a Laplace formulation. We want to proceed directly from the Laplace transform and to compute its inverse internally by a quadrature formula proposed by Stehfest [490].

The advantage of a Laplace formulation is that the numerical complexity of nonlinear systems can be reduced to a lower level. Linear differential equations, for example, can be transformed into algebraic equations and linear partial differential equations can be reduced to ordinary differential equations. The simplified systems are often solvable by analytical considerations.

Let us assume that the model function is given in form of a Laplace transform, say $H(p, s, c) \in \mathbb{R}^r$, depending on the parameter vector p to be fitted, the Laplace variable s , and an optional concentration parameter c . The inversion is performed numerically by the quadrature formula of Stehfest [490], for example. Proceeding from coefficients which can be evaluated before starting the parameter estimation algorithm, we compute the expression

$$h_k(p, t, c) = \frac{\ln 2}{t} \sum_{i=1}^{2q} v_i H_k(p, \frac{i \ln 2}{t}, c) \quad (2.17)$$

for $k = 1, \dots, r$. The vector-valued function $h(p, t, c)$ is a numerical approximation of the inverse Laplace transform of $H(p, s, c)$ subject to an accuracy given by the number q , and defines our fitting criterion. It is recommended to use q between 5 and 8. Any smaller value decreases the required accuracy, any larger value introduces additional round-off errors. However, numerical instabilities must be expected in case of highly oscillating functions.

A particular advantage of the above formula is that we get easily the gradient of the fitting function subject to the parameters to be estimated, if derivatives of the Laplace transform $\nabla_p H_k(p, s, c)$ are available,

$$\nabla_p h_k(p, t, c) = \frac{\ln 2}{t} \sum_{i=1}^{2q} v_i \nabla_p H_k(p, \frac{i \ln 2}{t}, c) . \quad (2.18)$$

Proceeding now from measurements (t_i, c_j, y_{ij}^k) and weights w_{ij}^k , $i = 1, \dots, l_t$, $j = 1, \dots, l_c$, and $k = 1, \dots, r$, we get the data fitting problem

$$p \in \mathbb{R}^n : \min_{p_l \leq p \leq p_u} \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, t_i, c_j) - y_{ij}^k))^2 . \quad (2.19)$$

General nonlinear constraints are omitted for simplicity.

Example 2.3 (LKIN_L3) *The formulation of a data fitting model in the Laplace space is illustrated by a simple test case, see also Example 2.2. A linear ordinary differential equation describes a kinetic process in the form*

$$\begin{aligned} \dot{y}_1 &= -k_{12}y_1 & , \quad y_1(0) &= D , \\ \dot{y}_2 &= k_{12}y_1 - k_{21}y_2 & , \quad y_2(0) &= 0 . \end{aligned}$$

q	$residual$	p_1^*	p_2^*
4	$0.212 \cdot 10^{-3}$	0.10015	0.049823
5	$0.105 \cdot 10^{-4}$	0.100120	0.049960
6	$0.165 \cdot 10^{-5}$	0.100081	0.049963
7	$0.104 \cdot 10^{-5}$	0.100067	0.049953
8	$0.101 \cdot 10^{-5}$	0.100063	0.049950
9	$0.101 \cdot 10^{-5}$	0.100062	0.049949
10	$0.101 \cdot 10^{-5}$	0.100060	0.049948
-	$0.462 \cdot 10^{-7}$	0.100054	0.050002

Table 2.2: Final Residuals and Solution Vectors

If Y_1 and Y_2 denote the Laplace transforms of y_1 and y_2 , respectively, and if we exploit the linearity of the Laplace operator, we get the system

$$\begin{aligned} sY_1 - D &= -k_{12}Y_1, \\ sY_2 &= k_{12}Y_1 - k_{21}Y_2. \end{aligned}$$

Let $Y_1(p, s, D)$ and $Y_2(p, s, D)$ be the solution of this system with $p = (k_{12}, k_{21})^T$, i.e.,

$$\begin{aligned} Y_1(p, s, D) &= \frac{D}{s + k_{12}}, \\ Y_2(p, s, D) &= \frac{k_{12}D}{(s + k_{12})(s + k_{21})}. \end{aligned}$$

The parameters to be estimated, are the transition coefficients k_{12} and k_{21} , and three values are given for the initial dose D . Experimental data are simulated in the following way. We proceed from the time values given in Example 2.2 and $p = (0.1, 0.05)^T$, compute exact solution values for $D = 50, 100, 150$, and add a small random noise in the order of 10^{-4} . Then we execute the least squares code DFNLP with termination accuracy 10^{-5} starting from $p^0 = (1.0, 0.1)^T$ for different values of q .

The numerical results are listed in Table 2.2, where the last line contains the results obtained for the exact solution. DFNLP converges within 21 iterations in all cases. We see that the residual is improved from $q = 4$ to $q = 10$, but numerical instabilities prevent further significant improvements for q larger than 7. The residual is scaled by the sum of squared measurement values for each measurement set.

2.4 Steady State Equations

We consider now parameter estimation problems where the fitting function depends on a variable t called *time*, a further independent model variable c called *concentration*, the parameter vector p to be estimated, and the solution z of a steady state system, i.e., a system of time-dependent nonlinear equations. Again, it is supposed that r measurement sets of the form

$$(t_i, c_j, y_{ij}^k), \quad i = 1, \dots, l_t, \quad j = 1, \dots, l_c, \quad k = 1, \dots, r, \quad (2.20)$$

are given with l_t time values, l_c concentration values, and $l = l_t l_c r$ corresponding measured experimental data.

Together with a fitting criterion function $h(p, z, t, c)$, we get a parameter estimation problem (2.1), (2.2), or (2.3) by

$$f_s(p) = w_{ij}^k (h_k(p, z(p, t_i, c_j), t_i, c_j) - y_{ij}^k), \quad (2.21)$$

where s runs from 1 to $l = l_t l_c r$ in any order. The state variable $z(p, t, c) \in \mathbb{R}^m$ is implicitly defined by the solution z of the system

$$\begin{aligned} s_1(p, z, t, c) &= 0, \\ &\dots \\ s_m(p, z, t, c) &= 0. \end{aligned} \quad (2.22)$$

The equations are often obtained by neglecting the transient part of a differential equation, so that the dynamical system is considered in the steady state.

The system functions are assumed to be continuously differentiable with respect to variables p and z . Moreover, we require the regularity of the system, i.e., that the system is solvable, and that the derivative matrix

$$\nabla_z s(p, z, t, c) = \left(\frac{\partial s_j(p, z, t, c)}{\partial z_i} \right)_{i=1, m; j=1, m} \quad (2.23)$$

has full rank for all p with $p_l \leq p \leq p_u$ and for all z , for which a solution $z(p, t, c)$ exists. Consequently, the function $z(p, t, c)$ is differentiable with respect to all p in the feasible domain.

Now let t be fixed and let $z(p, t, c)$ a solution of the system of equations. If we denote $s(p, z, t, c) = (s_1(p, z, t, c), \dots, s_m(p, z, t, c))^T$ for all x and z , we get from the identity $s(p, z(p, t, c), t, c) = 0$, which is to be satisfied for all p , the derivative

$$\nabla_p s(p, z(p, t, c), t, c) + \nabla z(p, t, c) \nabla_z s(p, z(p, t, c), t, c) = 0. \quad (2.24)$$

Here, $\nabla_p s(p, z, t, c)$ and $\nabla_z s(p, z, t, c)$ denote the Jacobian matrices of the vector-valued function $s(p, z, t, c)$ with respect to the parameters p and z , respectively. In other words, the desired Jacobian $\nabla z(p, t, c)$ is obtained by solving the linear system

$$\nabla_p s(p, z(p, t, c), t, c) + V \nabla_z s(p, z(p, t, c), t, c) = 0, \quad (2.25)$$

where V is a $m \times n$ -matrix. Note that we describe here the implicit function theorem. Since $\nabla_z s(p, z(p, t, c), t, c)$ is nonsingular, the above system is uniquely solvable.

Finally, we obtain the gradients of the fitting criterion from

$$\nabla f_s(p) = 2w_{ij}^k (\nabla_p h(p, z(p, t_i, c_j), t_i, c_j) + \nabla z(p, t_i, c_j) \nabla_z h(p, z(p, t_i, c_j), t_i, c_j)) \quad (2.26)$$

for $i = 1, \dots, l_t$, $j = 1, \dots, l_c$, and $k = 1, \dots, r$, where $z(p, t, c)$ is the solution of the system of nonlinear equations (2.22) and $\nabla z(p, t, c) = V$ computed from (2.25).

In addition, we allow any nonlinear restrictions on the parameters to be estimated, in form of general equality or inequality constraints

$$\begin{aligned} g_j(p) &= 0, \quad j = 1, \dots, m_e, \\ g_j(p) &\geq 0, \quad j = m_e + 1, \dots, m_r. \end{aligned} \quad (2.27)$$

It is assumed that all functions are continuously differentiable subject to p . The above formulation (2.27) includes also the possibility, to define dynamical inequality restrictions that are constraints depending on the state variable $z(p, t, c)$ at known time and concentration values. Thus, constraints of the form

$$g_j(p) = \bar{g}_j(p, z(p, t_{ij}, c_{kj}), t_{ij}, c_{kj}) \quad (2.28)$$

for $m_e < j \leq m_r$ are permitted at predetermined time and concentration values, that must coincide with some of the given independent measurement data. If constraints are to be defined independently from given measurement data, it is recommended to insert dummy experimental values with zero weights at the desired time and concentration points t_{ij} and c_{kj} , respectively. A more extensive discussion and an example is found in the subsequent section.

Example 2.4 (RECLIG19) *To illustrate the data fit of a steady state system, consider the following example that is similar to a receptor-ligand binding study with one receptor and two ligands, see Schittkowski [432]:*

$$\begin{aligned} z_1(1 + p_1 z_2 + p_2 z_3) - p_3 &= 0, \\ z_2(1 + p_1 z_1) - p_4 &= 0, \\ z_3(1 + p_2 z_1) - t &= 0. \end{aligned} \quad (2.29)$$

State variables are z_1 , z_2 , and z_3 , and the parameters to be fitted, are p_1 , p_2 , and p_4 . $p_3 = 100$ is fixed to avoid an overdetermined system, and t is the independent model variable. There is no concentration variable and the fitting criterion is $h(p, z, t) = p_4 - z_2$.

Experimental data are simulated at 13 time values 1, 5, 10, 50, 100, 500, ... 1,000,000 and subject to $p_1 = 0.01$, $p_2 = 0.0005$, and $p_4 = 1$. An error of 5 % is added to the measurement values. For our numerical tests, we use the starting values $z_1^0 = p_3$, $z_2^0 = p_4$, and $z_3^0 = t$ for solving the system of nonlinear equations. $p_1 = 0.1$, $p_2 = 0.0001$, and $p_4 = 2$ are the

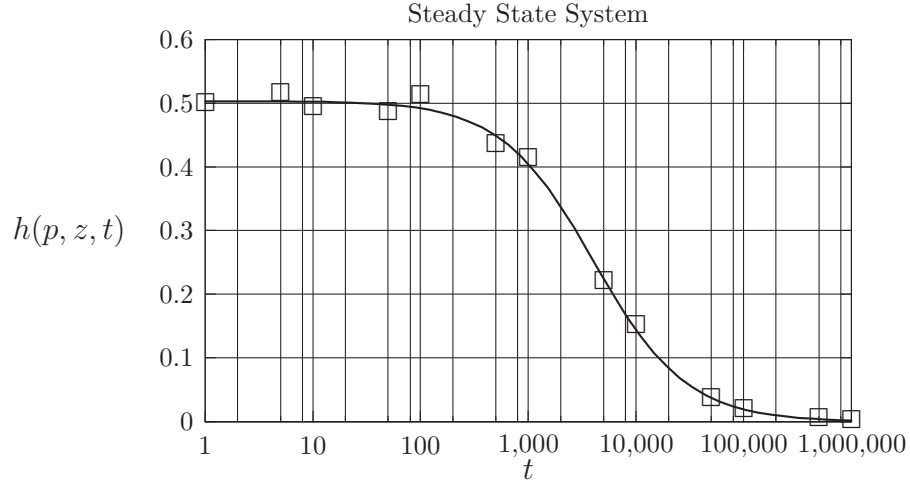


Figure 2.4: Model Function and Data Plot

starting values for the least squares algorithm *DFNLP* of Schittkowski [429] executed with a termination tolerance of 10^{-9} . After 11 iterations, we reach the parameters $p_1 = 0.0109$, $p_2 = 0.000554$, and $p_4 = 0.985$. The final residual is 0.00020. Model function values and simulated measurement data are plotted in Figure 2.4.

2.5 Ordinary Differential Equations

2.5.1 Standard Formulation

As before, we proceed from r data sets of the form

$$(t_i, c_j, y_{ij}^k), \quad i = 1, \dots, l_t, \quad j = 1, \dots, l_c, \quad k = 1, \dots, r, \quad (2.30)$$

where l_t *time* values, l_c *concentration* values and $l_t l_c r$ corresponding measurement values are defined. The vector-valued model function

$$h(p, y(p, t, c), t, c) = (h_1(p, y(p, t, c), t, c), \dots, h_r(p, y(p, t, c), t, c))^T$$

depends on the concentration parameter c and in addition on the solution $y(p, t, c)$ of a system of m ordinary differential equations with initial values,

$$\begin{aligned} \dot{y}_1 &= F_1(p, y, t, c) \quad , \quad y_1(0) = y_1^0(p, c) \quad , \\ &\dots \\ \dot{y}_m &= F_m(p, y, t, c) \quad , \quad y_m(0) = y_m^0(p, c) \quad . \end{aligned} \quad (2.31)$$

Without loss of generality, we assume that, as in many real life situations, the initial time is zero. The initial values of the differential equation system $y_1^0(p, c), \dots, y_m^0(p, c)$ may depend on one or more of the system parameters to be estimated, and on the concentration parameter c .

In this case, we have to assume in addition that the observation times are strictly increasing, and get the objective functions

$$\sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, y(p, t_i, c_j), t_i, c_j) - y_{ij}^k))^2 \quad (2.32)$$

for the least squares norm,

$$\sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} w_{ij}^k |h_k(p, y(p, t_i, c_j), t_i, c_j) - y_{ij}^k| \quad (2.33)$$

for the L_1 -norm, and

$$\max_{k=1, \dots, r; i=1, \dots, l_t; j=1, \dots, l_c} w_{ij}^k |h_k(p, y(p, t_i, c_j), t_i, c_j) - y_{ij}^k| \quad (2.34)$$

for the maximum-norm.

The system of ordinary differential equations is to be solved numerically by explicit or implicit integration methods. To be able to evaluate the gradient of the fitting criterion with respect to p , for example to compute

$$\nabla f_s(p) = 2w_{ij}^k (\nabla_p h(p, y(p, t_i, c_j), t_i, c_j) + \nabla y(p, t_i, c_j) \nabla_y h(p, y(p, t_i, c_j), t_i, c_j)) \quad (2.35)$$

in case of (2.32), for $i = 1, \dots, l_t$, $j = 1, \dots, l_c$, and $k = 1, \dots, r$, we need the derivatives of the solution vector $y(p, t, c)$ subject to p , that is $\nabla y(p, t_i, c_j)$. For evaluating $\nabla y(p, t, c)$, we apply either outer approximations, for example a forward difference or any similar formula, we add sensitivity equations to the ODE, or we use internal numerical differentiation.

Example 2.5 (LKIN_O3) *We consider again Example 2.2, now given in the notation*

$$\begin{aligned}\dot{y}_1 &= -k_{12}y_1 & , \quad y_1(0) = D & , \\ \dot{y}_2 &= k_{12}y_1 - k_{21}y_2 & , \quad y_2(0) = 0 & ,\end{aligned}$$

where three different initial doses $D_1 = 50$, $D_2 = 100$, and $D_3 = 150$ are applied. Experimental data are generated for the same 13 time values starting the simulation from $k_{12} = 0.1$ and $k_{21} = 0.05$ and adding a random error of 5 %. The differential equation is solved by an explicit integration algorithm with termination accuracy 10^{-10} and internal numerical differentiation. We get exactly the same solution after 26 iterations, as for the explicit formulation. Only the calculation time is about 50 times bigger because of the extremely accurate ODE solution, where more than 10 correct digits are required. If, on the other hand, we reduce the integration and optimization accuracy to 10^{-5} , we get the somewhat different solution $k_{12} = 0.10028$, $k_{21} = 0.04994$ again in 26 iterations, but now the calculation time is only 18 times bigger.

2.5.2 Differential Algebraic Equations

Now we add algebraic equations to the system of differential ones (2.31). In this case, the fitting criterion $h(p, y(p, t, c), z(p, t, c), t, c)$ depends on m_d differentiable variables $y(p, t, c)$ and m_a additional algebraic variables $z(p, t, c)$. The dynamical system is given in the form

$$\begin{aligned}\dot{y}_1 &= F_1(p, y, z, t, c) & , \quad y_1(0) = y_1^0(p, c) & , \\ &\dots & & \\ \dot{y}_{m_d} &= F_{m_d}(p, y, z, t, c) & , \quad y_{m_d}(0) = y_{m_d}^0(p, c) & , \\ 0 &= G_1(p, y, z, t, c) & , \quad z_1(0) = z_1^0(p, c) & , \\ &\dots & & \\ 0 &= G_{m_a}(p, y, z, t, c) & , \quad z_{m_a}(0) = z_{m_a}^0(p, c) & .\end{aligned}\tag{2.36}$$

Without loss of generality, we assume again that the initial time is zero. The initial values of the differential equations $y_1^0(p, c), \dots, y_{m_d}^0(p, c)$ and of the algebraic equations $z_1^0(p, c), \dots, z_{m_a}^0(p, c)$ may depend on the system parameters to be estimated, and on the concentration parameter c .

Now $y(p, t, c)$ and $z(p, t, c)$ are solution vectors of a joint system of $m_d + m_a$ differential and algebraic equations (DAE). The system is called an index-1-problem or an index-1-DAE, if the algebraic equations can be solved subject to z , i.e., if the matrix

$$\nabla_z G(p, y, z, t, c)\tag{2.37}$$

possesses full rank. If $m_d = 0$, we get a steady state system as discussed in Section 2.3. In all other cases, we obtain DAE's with a higher index, see for example Hairer and Wanner [199] for a suitable definition and more details. For simplicity, we consider now only problems of index one, although one of our standard implicit solver is able to solve also index-2- and index-3-problems. Note that problems with higher index can be transformed to problems of index one by successive differentiation of the algebraic equations.

We have to be very careful when defining the initial values of the model, since they must satisfy the consistency equation

$$G_1(p, y^0(p, c), z^0(p, c), t, c) = 0, \dots, G_{m_a}(p, y^0(p, c), z^0(p, c), t, c) = 0. \quad (2.38)$$

Otherwise, we have to check, whether the consistency condition is satisfied before starting the integration. If not, consistent initial values must be computed by solving the above system of nonlinear equations subject to z , where the initial values for the differential equations are inserted.

Example 2.6 (BATCHREA) *We consider a simplified batch reactor model discussed by Caracotsios and Stewart [75], where 6 differential and 4 algebraic equations are given,*

$$\begin{aligned} \dot{y}_1 &= -p_3 y_2 z_2 & , & \quad y_1(0) = 1.5776, \\ \dot{y}_2 &= -p_1 y_2 y_6 + p_2 z_4 - p_3 y_2 z_2 & , & \quad y_2(0) = 8.32, \\ \dot{y}_3 &= p_3 y_2 z_2 + p_4 y_4 y_6 - p_5 z_3 & , & \quad y_3(0) = 0, \\ \dot{y}_4 &= -p_4 y_4 y_6 + p_5 z_3 & , & \quad y_4(0) = 0, \\ \dot{y}_5 &= p_1 y_2 y_6 - p_2 z_4 & , & \quad y_5(0) = 0, \\ \dot{y}_6 &= -p_1 y_2 y_6 - p_4 y_4 y_6 + p_2 z_4 + p_5 z_3 & , & \quad y_6(0) = 0.0131, \\ 0 &= -z_1 - 0.0131 + y_6 + z_2 + z_3 + z_4 & , & \quad z_1(0) = z_1^0, \\ 0 &= (p_7 + z_1) z_2 - p_7 y_1 & , & \quad z_2(0) = z_1^0, \\ 0 &= (p_8 + z_1) z_3 - p_8 y_3 & , & \quad z_3(0) = 0, \\ 0 &= (p_6 + z_1) z_4 - p_6 y_5 & , & \quad z_4(0) = 0. \end{aligned} \quad (2.39)$$

$z_1^0 = 0.5 \left(-p_7 + \sqrt{p_7^2 + 4p_7 y_1(0)} \right)$ guarantees consistent initial values for the algebraic variables.

Measurement data are simulated for $t_i = i$, $i = 1, \dots, 10$, $p_i = 1$, $i = 1, \dots, 8$, with 3 correct digits, and fitting criteria are y_1, \dots, y_6 . The DAE is integrated by an implicit method with absolute and relative termination tolerance 10^{-5} , and the least squares code DFNLP is executed with final accuracy of 10^{-9} .

DFNLP terminates after 58 iterations with a scaled residual of 0.000029. Starting values p^0 and final parameter values p^* are shown in Table 2.3. The data fitting model is highly overdetermined. In other words, we have too many parameters making it impossible to evaluate them at least from the given data. Only some of them are in the order of the known exact values.

i	p^0	p^*
1	3.0	1.031
2	1.5	0.751
3	1.1	0.591
4	0.5	0.946
5	0.6	1.029
6	1.4	2.983
7	10.0	9.752
8	0.1	1.068

Table 2.3: Starting and Computed Values for Example 2.6

2.5.3 Switching Points

There are many practical situations, where model equations change during the integration over the time variable, and where corresponding initial values at the switching points must be adopted. A typical example is a pharmacokinetic application with an initial infusion and subsequent application of drug doses by injection. It is even possible in these cases that the solution becomes non-continuous at a switching respectively break point.

We assume for simplicity that the dynamical model is given in form of an ordinary differential equation with initial conditions. In a similar way, we may define switching points for steady state systems or differential algebraic equations. We describe the model by the equations

$$\begin{aligned} \dot{y}_1^0 &= F_1^0(p, y^0, t, c) \quad , \quad y_1^0(0) = \bar{y}_1^0(p, c) \quad , \\ &\dots \\ \dot{y}_m^0 &= F_m^0(p, y^0, t, c) \quad , \quad y_m^0(0) = \bar{y}_m^0(p, c) \quad , \end{aligned} \tag{2.40}$$

for $0 \leq t \leq \tau_1$ and

$$\begin{aligned} \dot{y}_1^i &= F_1^i(p, y^i, t, c) \quad , \quad y_1^i(\tau_i) = \bar{y}_1^i(p, c, y_1^{i-1}(p, \tau_i, c)) \quad , \\ &\dots \\ \dot{y}_m^i &= F_m^i(p, y^i, t, c) \quad , \quad y_m^i(\tau_i) = \bar{y}_m^i(p, c, y_m^{i-1}(p, \tau_i, c)) \end{aligned} \tag{2.41}$$

for $\tau_i \leq t \leq \tau_{i+1}$, $i = 1, \dots, n_b$. n_b is the number of break respectively switching points τ_i with $0 < \tau_1 < \dots < \tau_{n_b} < T$, where $\tau_{n_b+1} = T$ is the last experimental time. The initial values of each subsystem are given by functions $\bar{y}_j^i(p, c, y)$ depending on the parameters to be estimated, the actual concentration value, and the solution of the previous interval at the break point τ_i . Internally the integration of the differential equation is restarted at a switching point.

Example 2.7 (LKIN_BR) Consider the linear compartment model of Example 2.2,

$$\begin{aligned} \dot{y}_1 &= -k_{12}y_1 \quad , \quad y_1(0) = D_0 \quad , \\ \dot{y}_2 &= k_{12}y_1 - k_{21}y_2 \quad , \quad y_2(0) = 0 \end{aligned} \tag{2.42}$$

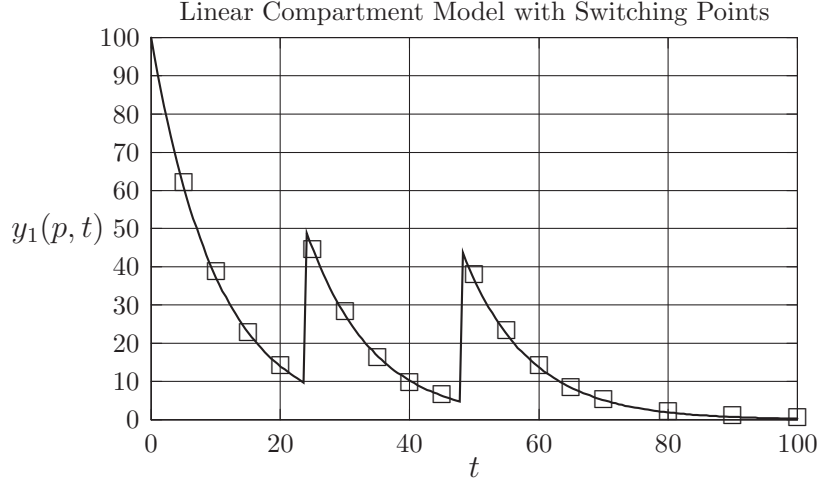


Figure 2.5: Function and Data Plot for Compartment 1

with an initial dose $D_0 = 100$ for the input compartment y_1 . After $\tau_1 = 24$ respectively $\tau_2 = 48$ time units, another dose of $D_1 = 40$ respectively $D_2 = 40$ is applied. Formally, the initial values at $t = 0$ and the switching times are given by

$$\begin{aligned}\bar{y}_1^0(p) &= D_0 , \\ \bar{y}_2^0(p) &= 0 , \\ \bar{y}_1^1(p, y_1^0(p, \tau_1)) &= y_1^0(p, \tau_1) + D_1 , \\ \bar{y}_2^1(p, y_2^0(p, \tau_1)) &= y_2^0(p, \tau_1) , \\ \bar{y}_1^2(p, y_1^1(p, \tau_2)) &= y_1^1(p, \tau_2) + D_2 , \\ \bar{y}_2^2(p, y_2^1(p, \tau_2)) &= y_2^1(p, \tau_2) .\end{aligned}$$

Here we omit the concentration variable c to simplify the notation, and the parameter vector is $p = (k_{12}, k_{21})^T$.

Experimental data are simulated for 17 time values between 0 and 100 and $k_{12} = 0.1$ and $k_{21} = 0.05$. A random error of 5 % is added to the obtained data. The differential equation is integrated by an explicit Runge-Kutta code with absolute and relative termination accuracy 10^{-7} . The least squares solver DFNLP is started at $k_{12} = 1$ and $k_{21} = 1$ with a termination tolerance of 10^{-10} . After 60 iterations the termination conditions are satisfied at $k_{12} = 0.0984$ and $k_{21} = 0.0512$. Function and data plots are shown in Figures 2.5 and 2.6.

It is even possible that break points become variables that are to be adapted during the optimization process. However, there is a dangerous situation, if during an optimization run a variable switching point passes or approximates an experimental time value. If both

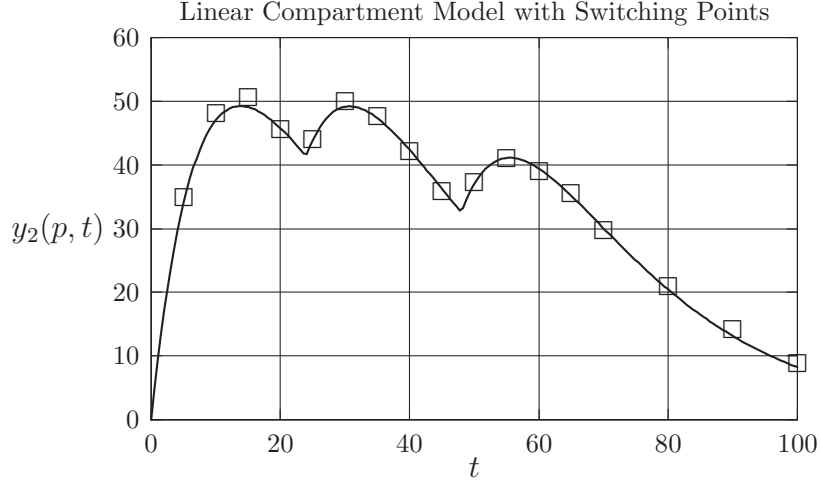


Figure 2.6: Function and Data Plot for Compartment 2

coincide and if there is a non-continuous transition, then the underlying model function is no longer differentiable subject to the parameter to be optimized. Possible reactions of the least squares algorithm are slow final convergence rates or break down because of internal numerical difficulties. On the other hand, variable switching points are highly valuable when trying to model for instance the input feed of chemical or biological processes given by a bang-bang control function or any other one with variable break points.

To give an example, consider a pharmacokinetic model with an initial lag time, that is unknown a priori.

Example 2.8 (LKIN_LA) *We consider the same linear compartment model as before, but now with a lag time τ ,*

$$\begin{aligned} \dot{y}_1 &= \begin{cases} 0 & , \text{ if } t < \tau , \\ -k_{12}y_1 & , \text{ if } t \geq \tau , \end{cases} \quad , \quad y_1(0) = D_0 \quad , \\ \dot{y}_2 &= k_{12}y_1 - k_{21}y_2 \quad , \quad y_2(0) = 0 \end{aligned} \quad (2.43)$$

with an initial dose $D_0 = 100$. Experimental data are simulated under the same conditions as for Example 2.7, but now with $\tau = 5$ as additional model parameter to be estimated, and without further switching points. DFNLP is started at $\tau = 1$, $k_{12} = 1$, and $k_{21} = 1$ with a termination tolerance of 10^{-10} . After 59 iterations, the termination conditions are satisfied at $\tau = 5.21$, $k_{12} = 0.1022$, and $k_{21} = 0.0499$. Function and data plots are shown in Figures 2.7 and 2.8.

So far we considered only given switching points, which are stated explicitly as part of the model formulation. However, there are very many other reasons for discontinuities of

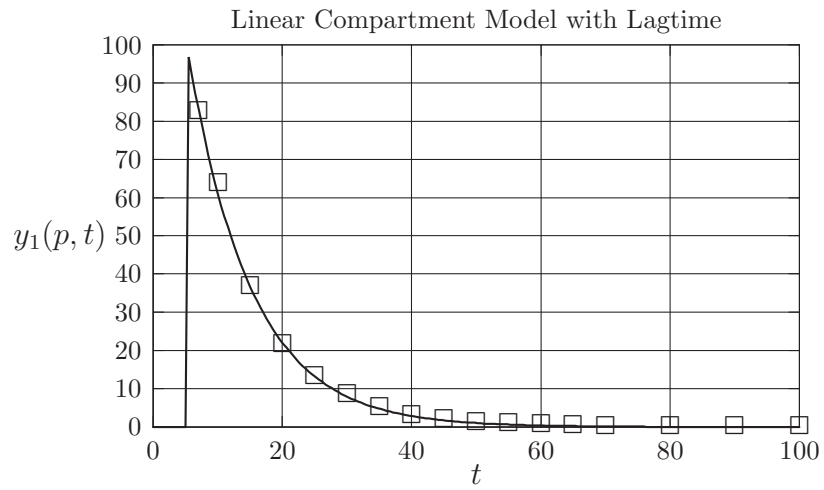


Figure 2.7: Function and Data Plot for Compartment 1

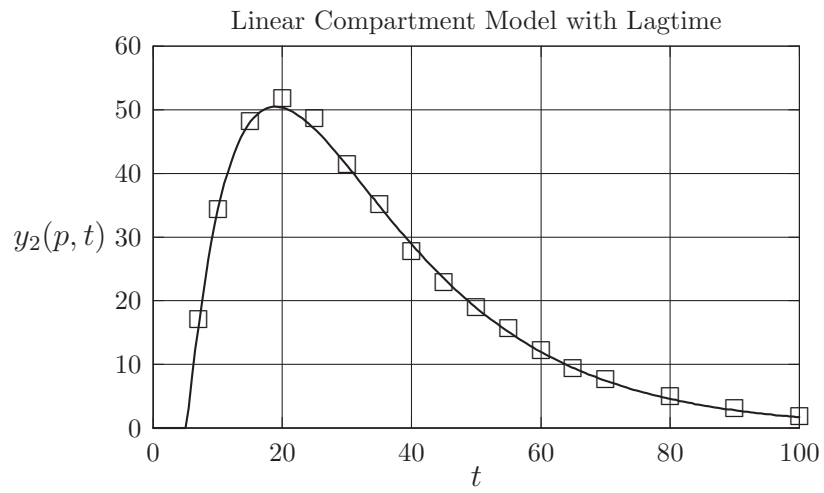


Figure 2.8: Function and Data Plot for Compartment 2

the right-hand side of a system of ordinary differential equations, where switching points are defined implicitly. They appear especially in chemical engineering or multibody systems, see Preston and Berzins [390] in the first and Eich-Soellner and Führer [132] in the second case. A typical example is dry friction between two bodies, see Example 2.10 below.

It must be expected that the direct integration of an ODE with discontinuities leads to numerical instabilities, since very small stepsizes must be used to pass around a *corner* in the solution. Typically, the ODE formulation is extended by a vector-valued switching function

$$q(p, y(p, t), t) = (q_1(p, y(p, t), t), \dots, q_{n_b}(p, y(p, t), t))^T ,$$

which must be given by a user a priori to specify the change of a sign in the model equations for example, and which could also depend on parameters to be estimated. Here we omit the additional concentration variable c to simplify the notation. Then we proceed from the dynamical system

$$\begin{aligned} \dot{y}_1 &= F_1(p, y, t, \text{sign}(q(p, y, t))) \quad , \quad y_1(0) = y_1^0(p) \quad , \\ &\quad \dots \\ \dot{y}_m &= F_m(p, y, t, \text{sign}(q(p, y, t))) \quad , \quad y_m(0) = y_m^0(p) \quad . \end{aligned} \tag{2.44}$$

Example 2.9 (LKIN_LA) Consider again Example 2.8. We define $q(p, y, t) = t - \tau$ and

$$\begin{aligned} F_1(p, y, t, \text{sign}(q(p, y, t))) &= \begin{cases} 0 & , \text{ if } \text{sign}(q) = -1 \quad , \\ -k_{12}y_1 & , \text{ if } \text{sign}(q) = +1 \quad , \end{cases} \quad , \quad y_1(0) = D_0 \quad , \\ F_2(p, y, t, \text{sign}(q(p, y, t))) &= \begin{cases} k_{12}y_1 - k_{21}y_2 & , \end{cases} \quad , \quad y_2(0) = 0 \end{aligned} \tag{2.45}$$

with $p = (k_{12}, k_{21}, \tau)^T$.

The implicitly given switching times must be computed internally during the numerical integration of (2.44). As soon as a change of the sign of the switching function $q(p, y, t)$ is observed, a special root finding sub-algorithm must be started to locate the switching time, leading to substantial additional numerical efforts. The integration is then restarted from the computed value, see Eich-Soellner and Führer [132] for more details, or Chartres and Stepleman [86], Mannshardt [317], Carver [80], Ellison [136], or Gear and Osterby [165] for alternative approaches.

In some situations, however, it is possible to avoid the internal approximation of discontinuities, by introducing artificial switching times that must be optimized together with the given parameters p . The switching function $q(p, y, t)$ can be avoided completely, and the integration is safely restarted at the known switching times without crossing a discontinuity.

To apply the proposed strategy, we need to know, how to replace the switching function $q(p, y, t)$ by suitable switching times, and one should know a bit about the distribution of switching times, for example their number and serial order. But if we are able to collect some information a priori, it is possible to simplify and stabilize the numerical integration, as shown by the subsequent example.

Example 2.10 (DRY_FRI1/2/3) We consider a simple mass oscillator with dry friction between two bodies, confer Eich-Soellner and Führer [132]. The dynamical system is given by two second-order differential equations

$$\begin{aligned} m_1 \ddot{x}_1 &= f_1 - \mu \operatorname{sign}(\dot{x}_1 - \dot{x}_2) \quad , \quad x_1(0) = 1 \quad , \quad \dot{x}_1(0) = 0 \quad , \\ m_2 \ddot{x}_2 &= f_2 + \mu \operatorname{sign}(\dot{x}_1 - \dot{x}_2) \quad , \quad x_2(0) = 1 \quad , \quad \dot{x}_2(0) = 0 \quad , \end{aligned} \quad (2.46)$$

with $m_1 = m_2 = 1$, $f_1 = \sin t$, and $f_2 = 0$. $p = \mu$ is considered the unknown parameter to be estimated. When integrating the above system for $\mu = 1.5$ from $t = 0$ to $t = 10$ without any safeguards, we get an unstable solution as shown in Figure 2.9 only for $x_1(t)$. Numerical instabilities occur also for $x_2(t)$, $\dot{x}_1(t)$, and $\dot{x}_2(t)$. However, when reducing the influence of the friction coefficient to $\mu = 0.01$, we are able to integrate the system despite of the discontinuity, see Figures 2.10 and 2.11. The dotted lines represent $x_2(t)$ and $\dot{x}_2(t)$, respectively.

We generate artificial exact measurements for $x_1(t)$, $x_2(t)$, $\dot{x}_1(t)$, and $\dot{x}_2(t)$ at time values $t_i = i$ for $i = 1, \dots, 10$, and $p^* = \mu^* = 0.01$ without any random errors. Although the initial residual is in the order of 10^{16} , we are nevertheless able to recompute this optimal solution when starting from $p_0 = 1.5$, see Table 2.4, case 1. The least squares code DFNLP is applied with termination tolerance 10^{-13} .

From the differences of the velocities in Figure 2.11, it is obvious that we do not have more than two switching times. Thus, we add two additional optimization variables τ_1 and τ_2 , leading to the differential equations

$$\begin{aligned} m_1 \dot{x}_1^1 &= f_1 - \mu(\dot{x}_1^1 - \dot{x}_2^1) \quad , \quad x_1^1(0) = 1 \quad , \quad \dot{x}_1^1(0) = 0 \quad , \\ m_2 \dot{x}_2^1 &= f_2 + \mu(\dot{x}_1^1 - \dot{x}_2^1) \quad , \quad x_2^1(0) = 1 \quad , \quad \dot{x}_2^1(0) = 0 \end{aligned} \quad (2.47)$$

for all $t \leq \tau_1$,

$$\begin{aligned} m_1 \dot{x}_1^2 &= f_1 + \mu(\dot{x}_1^2 - \dot{x}_2^2) \quad , \quad x_1^2(0) = x_1^1(p, \tau_1) \quad , \quad \dot{x}_1^2(0) = \dot{x}_1^1(p, \tau_1) \quad , \\ m_2 \dot{x}_2^2 &= f_2 - \mu(\dot{x}_1^2 - \dot{x}_2^2) \quad , \quad x_2^2(0) = x_2^1(p, \tau_1) \quad , \quad \dot{x}_2^2(0) = \dot{x}_2^1(p, \tau_1) \end{aligned} \quad (2.48)$$

for all $\tau_1 \leq t \leq \tau_2$, and

$$\begin{aligned} m_1 \dot{x}_1^3 &= f_1 - \mu(\dot{x}_1^3 - \dot{x}_2^3) \quad , \quad x_1^3(0) = x_2^1(p, \tau_2) \quad , \quad \dot{x}_1^3(0) = \dot{x}_1^2(p, \tau_2) \quad , \\ m_2 \dot{x}_2^3 &= f_2 + \mu(\dot{x}_1^3 - \dot{x}_2^3) \quad , \quad x_2^3(0) = x_2^2(p, \tau_2) \quad , \quad \dot{x}_2^3(0) = \dot{x}_2^2(p, \tau_2) \end{aligned} \quad (2.49)$$

for all $t \geq \tau_2$. Now, the numerical instability for large friction coefficients mentioned above, is avoided, see Figures 2.12 and 2.13, where the estimates $\tau_1 = 6$ and $\tau_2 = 7$ are inserted for the switching times.

When starting DFNLP from $\mu = 1.5$, $\tau_1 = 6$, and $\tau_2 = 7$ we obtain the results of Table 2.4, case 2. We are able to identify the friction coefficient and the implicitly given switching times. It is necessary to add linear inequality constraints to satisfy $0 \leq \tau_1 \leq \tau_2 \leq 10$.

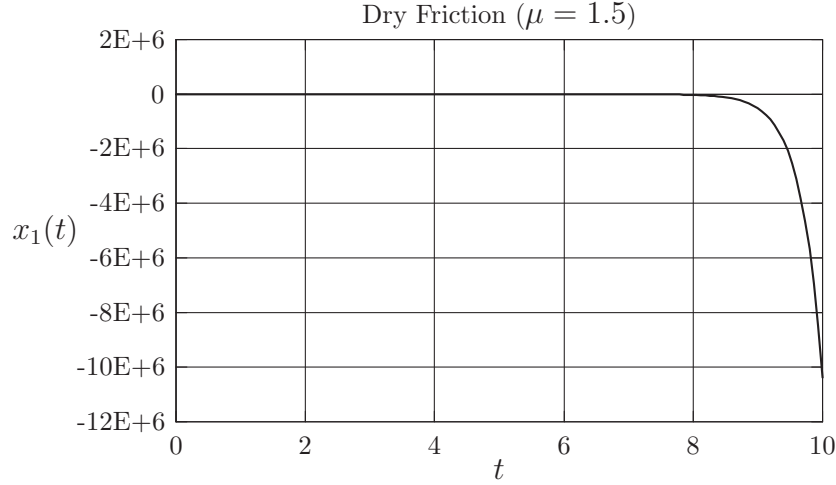


Figure 2.9: Function Plot for $x_1(t)$

case	n_{it}	residual	μ	τ_1	τ_2	τ_3	τ_4
1	40	$0.31 \cdot 10^{-8}$	0.009998899	-	-	-	-
2	57	$0.57 \cdot 10^{-11}$	0.009999998	5.78997	6.77911	-	-
3	33	$0.62 \cdot 10^{-12}$	0.010000005	3.97642	3.97642	5.79262	6.77731

Table 2.4: Number of Iterations, Residuals, and Optimal Solution for Dry Friction Problem

We cannot expect in general that the exact number of switching times is known. If, however, too many switching times are defined in form of additional optimization variables, there should be an overlay of final optimal values cancelling their influence. To illustrate this situation, we add two further switching times to our example, in the same way as outlined before. It is essential, to add linear inequality constraints of the form $0 \leq \tau_1 \leq \tau_2 \leq \tau_3 \leq \tau_4 \leq 10$ to the least squares optimization problem, to guarantee consistency of the model equations. When starting DFNLP with the same termination tolerance used before and the initial values $\mu = 1.5$, $\tau_1 = 4$, $\tau_2 = 5$, $\tau_3 = 6$, and $\tau_4 = 7$, we obtain the numerical results shown in Table 2.11, case 3. In all three cases, the differential equation is integrated by an implicit method with absolute and relative termination tolerance 10^{-5} .

The example shows that the parameter estimation problem is solved more efficiently and more accurately when introducing switching times. There is, however, a drawback of the proposed approach when too many switching times are defined. If some of them are redundant and become identical at an optimal solution as for case 3 of Example 2.10, then the final optimal solution is not unique and a slow final convergence speed must be expected.

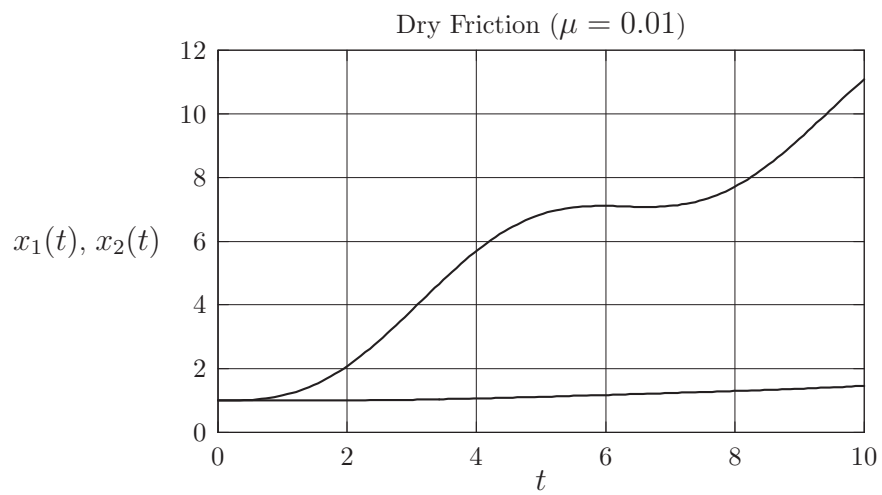


Figure 2.10: Function Plot for $x_1(t)$ and $x_2(t)$

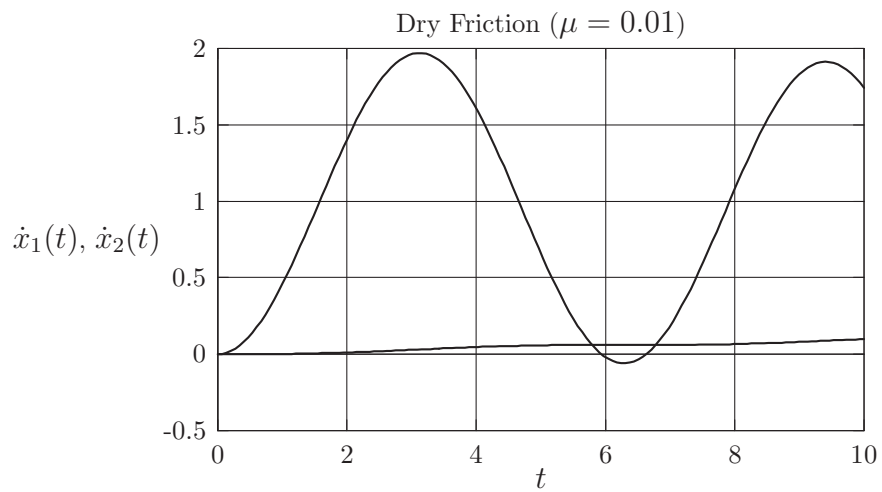


Figure 2.11: Function Plot for $\dot{x}_1(t)$ and $\dot{x}_2(t)$

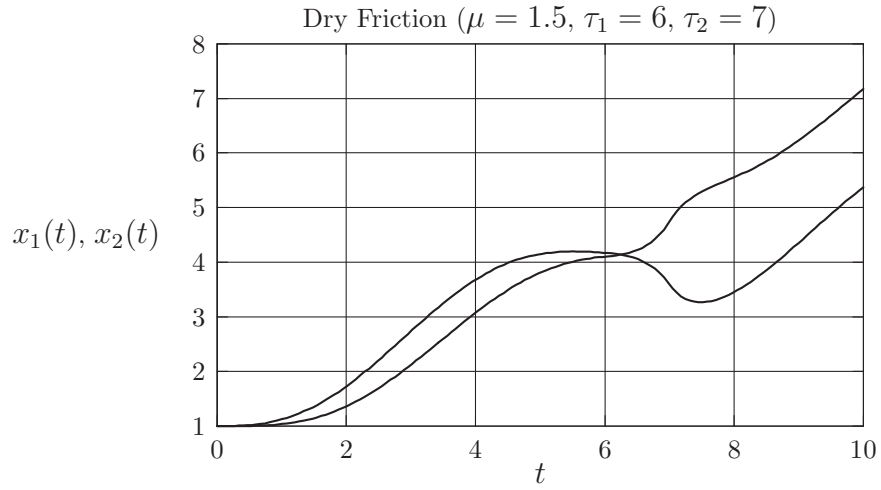


Figure 2.12: Function Plot for $x_1(t)$ and $x_2(t)$ with Two Switching Times

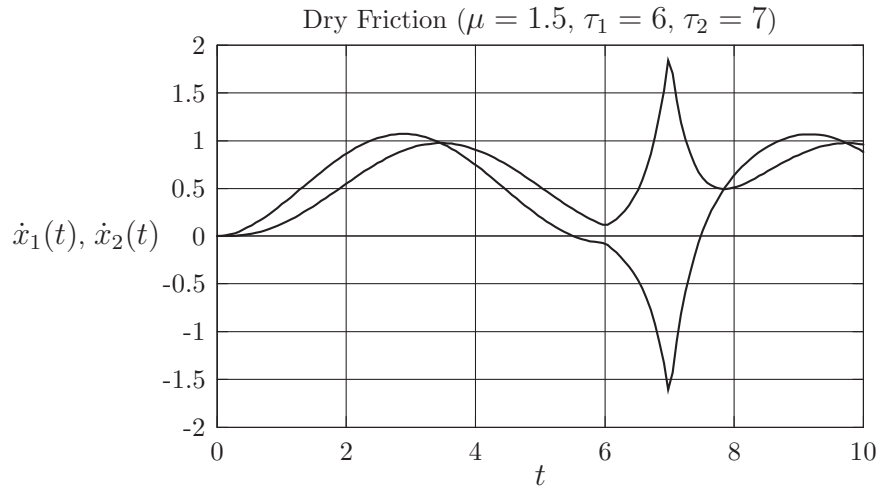


Figure 2.13: Function Plot for $\dot{x}_1(t)$ and $\dot{x}_2(t)$ with Two Switching Times

2.5.4 Constraints

Constraints in equality or inequality form

$$\begin{aligned} g_j(p) &= 0, \quad j = 1, \dots, m_e, \\ g_j(p) &\geq 0, \quad j = m_e + 1, \dots, m_r \end{aligned} \quad (2.50)$$

can be added to the general objective functions (2.1), (2.2), and (2.3), or to the data fitting formulations (2.32), (2.33), and (2.34), respectively. These restrictions define certain additional conditions to be satisfied for the parameters to be estimated. Functions $g_1(p), \dots, g_{m_r}(p)$ must be continuously differentiable everywhere in the \mathbb{R}^n .

A typical situation is discussed in Example 2.1, where the exact satisfaction of the first and last fit are required. The generation of more systematically introduced constraints will be discussed in subsequent sections.

Another frequent situation arises, if for example the parameters to be estimated, describe certain concentrations or fractions of a given amount of mass distribution, so that the sum over all parameters must be one at least at an optimal solution.

Example 2.11 (POPUL) *We consider population dynamics of 10 species with a given maximum population rate, each described by an ordinary differential equation*

$$\dot{y}_i = \alpha_i y_i (y_{\max} - y_i) - \beta_i y_i \quad (2.51)$$

for $i = 1, \dots, 10$ with $y_{\max} = 200$. Initial values are

$$y_i^0 = p_i N_0, \quad$$

where p_i is an unknown fraction of the initial population of $N_0 = 1,000$ individuals for all species, $i = 1, \dots, 10$. We are able to measure the total number of individual members of the population, i.e. fitting criterion is

$$h(p, y(p, t), t) = \sum_{i=1}^{10} y_i(p, t) \quad .$$

There is no additional concentration parameter and only one measurement set. Since each parameter p_i stands only for a fraction of a given constant value, the single equality constraint is

$$g(p) = \sum_{i=1}^{10} p_i - 1 = 0 \quad .$$

We declare suitable lower and upper bounds $0 \leq p_i \leq 1$ for the parameters to be estimated, $i = 1, \dots, 10$. Experimental data are simulated subject to a parameter vector $p^* \in \mathbb{R}^{10}$ and 83 equidistant time values 0.005, 0.01, 0.015, ..., 0.4. An error of 5 % is added to the

i	α_i	β_i	p_i^0	p_i^*	\bar{p}_i
1	0.05	0.05	0.2	0.20	0.1717
2	0.05	0.30	0.2	0.18	0.1715
3	0.05	0.60	0.2	0.17	0.1713
4	0.05	1.00	0.2	0.15	0.1711
5	0.60	0.05	0.2	0.12	0.0760
6	0.60	0.30	0.1	0.08	0.0765
7	0.60	0.60	0.1	0.05	0.0763
8	0.60	1.00	0.1	0.03	0.0761
9	1.20	0.05	0.1	0.01	0.0048
10	1.20	0.30	0.1	0.01	0.0049

Table 2.5: Constants and Staring, Optimal, and Computed Parameter Values for Population Dynamics

computed simulation data. The remaining constants α_i and β_i , the starting value p^0 , the optimal parameter p^* , and the computed parameter \bar{p} are summarized in Table 2.5. Numerical results are obtained by an explicit Runge-Kutta method with absolute and relative termination accuracy 10^{-7} . The least squares solver DFNLP is started with a termination tolerance of 10^{-10} . After 18 iterations the termination conditions are satisfied at \bar{p} with a residual value of 0.00093 scaled by sum of squares of measurement values. Initially, the equality constraint is violated, but is satisfied on termination subject to an error of $0.26 \cdot 10^{-13}$. A function and data plot is shown in Figure 2.14. Obviously, we are unable to recompute the known exact solution vector p^* . One possible reason is the we generated too large errors in the measurements.

In addition, the constraint functions may depend on the solution of the dynamical system at predetermined time and concentration values, i.e.

$$g_j(p) = \bar{g}_j(p, y(p, t_{i_j}, c_{k_j}), z(p, t_{i_j}, c_{k_j}), t_{i_j}, c_{k_j}) \quad (2.52)$$

for any j , $m_e < j \leq m_r$, where $y(p, t, c)$ and $z(p, t, c)$ denote the solution of a differential algebraic equation (2.36) depending on the parameter vector p to be estimated, and certain time and concentration values t and c , respectively. Note that dynamical constraints should be defined only in form of inequalities. Equality conditions can be handled as algebraic equations, and are part of the dynamical model.

The predetermined time and, if available at all, concentration values must coincide with some of the given experimental data. If constraints are to be defined independently from given measurement data, it is recommended to insert dummy experimental values with zero weights at the desired time and concentration points t_{i_j} and c_{k_j} , respectively, $j = m_e + 1, \dots, m_r$.

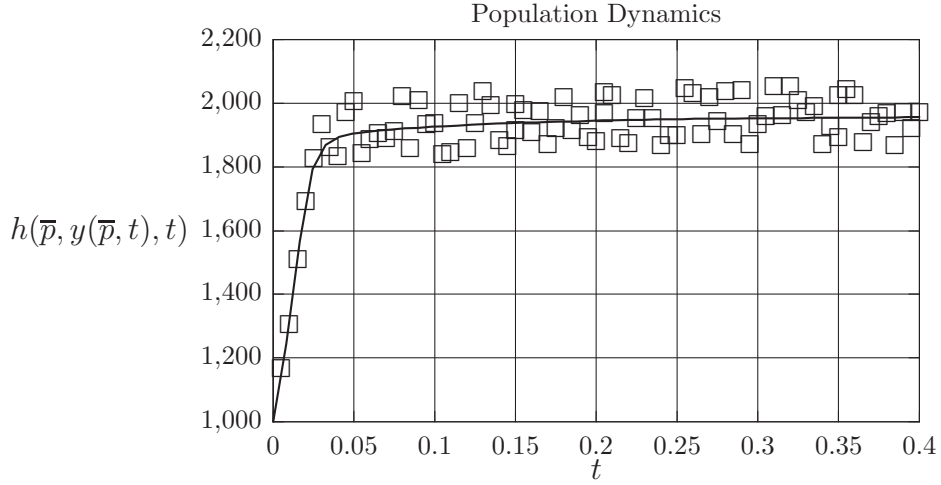


Figure 2.14: Function and Data Plot

Example 2.12 (LKIN_RE) We consider again Example 2.2 given by

$$\begin{aligned} \dot{y}_1 &= -p_1 y_1 & , \quad y_1(0) &= p_3 \quad , \\ \dot{y}_2 &= p_1 y_1 - p_2 y_2 & , \quad y_2(0) &= 0 \quad . \end{aligned}$$

Experimental data are shown in Table 2.6, where we added artificial measurements with zero weights for being able to define constraints of the form

$$g_j(p) = 45 - y_2(t_{j+4}) \geq 0$$

for $j = 1, \dots, 21$. In other words, we would like to limit the second state variable by the value 45.0 and require this condition at least at some discrete time values.

The differential equation is solved by an explicit integration algorithm with termination accuracy 10^{-6} and internal numerical differentiation. The least squares code DFNLP terminates after 10 iterations. Constraint 13 becomes active, and termination conditions are satisfied subject to a tolerance of 10^{-7} . Figure 2.15 shows the bounded state variable y_2 , where y_1 fits the data as in the unconstrained case.

2.5.5 Shooting Method

The traditional initial value approach discussed so far, breaks down, if we are unable to integrate the differential equation from initial time zero to the last experimental time t_{l_t} . One possible reason is numerical instability of the ODE that prevents a successful integration over the total interval.

i	t_i	y_i^1	w_i^1	y_i^2	w_i^2
1	1	85.34	1	10.86	1
2	2	79.44	1	21.1	1
3	3	79.13	1	27.01	1
4	4	71.18	1	34.45	1
5	5	56.2	1	36.41	1
6	6	0.0	0	0.0	0
7	7	0.0	0	0.0	0
8	8	0.0	0	0.0	0
9	9	0.0	0	0.0	0
10	10	29.66	1	46.27	1
11	11	0.0	0	0.0	0
12	12	0.0	0	0.0	0
13	13	0.0	0	0.0	0
14	14	0.0	0	0.0	0
15	15	17.47	1	48.73	1
16	16	0.0	0	0.0	0
17	17	0.0	0	0.0	0
18	18	0.0	0	0.0	0
19	19	0.0	0	0.0	0
20	20	10.8	1	48.17	1
21	21	0.0	0	0.0	0
22	22	0.0	0	0.0	0
23	23	0.0	0	0.0	0
24	24	0.0	0	0.0	0
25	25	5.81	1	41.76	1
26	30	3.23	1	27.32	1
27	40	0.93	1	20.7	1
28	50	0.32	1	11.18	1
29	60	0.1	1	5.76	1

Table 2.6: Experimental Data for Linear Kinetics Model

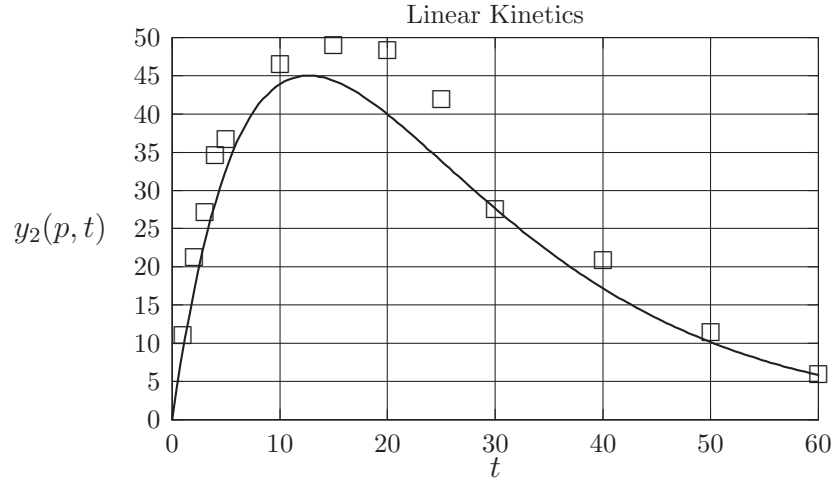


Figure 2.15: Function and Data Plot for Constrained State Variable

Example 2.13 (DEGEN) We consider a simple second order differential equation given by Bulirsch [63],

$$\ddot{y} = \mu^2 y - (\mu^2 + p^2) \sin(pt) \quad (2.53)$$

with initial values $\dot{y}(0) = 0$ and $y(0) = p$. The equivalent first order system is

$$\begin{aligned} \dot{y}_1 &= y_2, \\ \dot{y}_2 &= \mu^2 y_1 - (\mu^2 + p^2) \sin(pt). \end{aligned} \quad (2.54)$$

If we try to integrate (2.53) or (2.54) by any of the highly robust and efficient methods discussed in the previous chapter, from 0 to 1 with $\mu = 50$ and $p = 3.1415926535$, we get the result displayed in Figure 2.16. Obviously, the numerical solution fails because of internal instability of the equation, even if we start with a very low integration accuracy, say 10^{-10} . To explain the instability, we consider the general solution

$$\begin{aligned} y_1(t) &= \sin(pt) + \epsilon \sinh(\mu t), \\ y_2(t) &= p \cos(pt) + \epsilon \mu \cosh(\mu t) \end{aligned}$$

with $\epsilon = (\pi - p)/\mu$. Round-off errors and slight numerical deviations of p from the true π -value lead to an exponential increase of the computed solution because of the hyperbolic functions involved.

Other reasons for considering the multiple shooting approach are singularities preventing an integration over the whole interval, or highly oscillating solutions, where the initial trajectories for starting the least squares algorithm are far away from the experimental data.

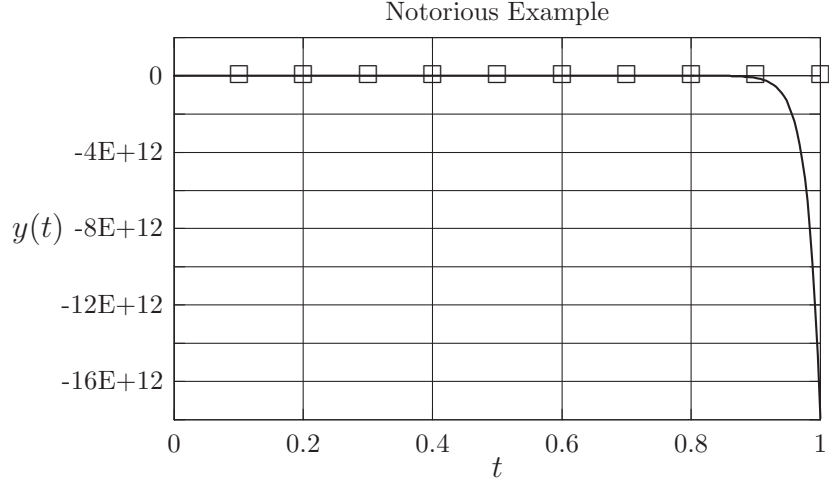


Figure 2.16: Single Shooting

Multiple shooting was first developed for solving boundary value and optimal control problems, see Bulirsch [63], Bock [51], or Deuffhard [111]. A brief outline is also found in Ascher and Petzold [11], see also Ascher, Mattheij, Russel [10] or Mattheij and Molenaar [324]. Multiple shooting is applied to the solution of data fitting problems by Bock [52].

The basic idea is to introduce n_s additional *shooting* points along the time axis, say

$$0 < \tau_1 < \dots < \tau_{n_s} < T , \quad (2.55)$$

where $T = t_{l_t}$ is the last experimental time value. For formal reasons, we define $\tau_0 = 0$. Integration is performed only from one shooting point τ_{i-1} to the next one τ_i , and then initialized with a shooting variable $s_i \in \mathbb{R}^m$, $i = 1, \dots, n_s$. m denotes the number of differential equations. The differences of shooting variables and solution at right-end of the previous shooting interval lead to additional nonlinear equality constraints.

In a more formal way, we proceed from the following system of m ordinary differential equations, where we omit the concentration variable for simplicity,

$$\begin{aligned} \dot{y}_1^0 &= F_1^0(p, y^0, t) \quad , \quad y_1^0(0) = \bar{y}_1^0(p) \quad , \\ &\dots \\ \dot{y}_m^0 &= F_m^0(p, y^0, t) \quad , \quad y_m^0(0) = \bar{y}_m^0(p) \quad , \end{aligned} \quad (2.56)$$

for $0 \leq t \leq \tau_1$, $\bar{y}_j^0(p)$ given initial values, $j = 1, \dots, m$, and

$$\begin{aligned} \dot{y}_1^i &= F_1^i(p, y^i, t) \quad , \quad y_1^i(\tau_i) = s_1^i \quad , \\ &\dots \\ \dot{y}_m^i &= F_m^i(p, y^i, t) \quad , \quad y_m^i(\tau_i) = s_m^i \end{aligned} \quad (2.57)$$

for $\tau_i \leq t \leq \tau_{i+1}$, $i = 1, \dots, n_s$, $\tau_{n_s+1} = T$.

This formulation is quite similar to (2.40) and (2.41), where switching points are taken into account. The difference is the treatment of initial values for restarting the integration. In (2.40) and (2.41) these values are known a priori as part of the mathematical model. Now the initial values are unknown parameters to be computed in addition to the model parameters p . Thus, we get mn_s artificial optimization parameters s_1, \dots, s_{n_s} , leading to the total set of optimization parameters

$$\bar{p} = (p_1, \dots, p_n, s_1^1, \dots, s_m^1, \dots, s_1^{n_s}, \dots, s_m^{n_s})^T .$$

One has to guarantee that the differences between the trajectories at their right end points coincide with the artificially introduced initial values for the subsequent interval. Thus, we get an additional set of mn_s nonlinear equality constraints

$$\begin{aligned} y^0(p, \tau_1) - s_1 &= 0 , \\ y^1(p, \tau_2) - s_2 &= 0 , \\ &\dots \\ y^{n_s-1}(p, \tau_{n_s}) - s_{n_s} &= 0 , \end{aligned} \tag{2.58}$$

where $y^0(p, t)$ denotes the solution of (2.56) and $y^i(p, t)$ the solution of (2.57) for $\tau_i \leq t \leq \tau_{i+1}$, $i = 1, \dots, n_s$.

Example 2.14 (LOT_VOL2) *A famous biological model describes the behaviour of a predator and a prey species of an ecological system, used as a standard parameter estimation test problem in the literature, cf. Clark [93], Varah [522], or Edsberg and Wedin [130]. The so-called Lotka-Volterra system consists of two equations*

$$\begin{aligned} \dot{y}_1 &= -k_1 y_1 + k_2 y_1 y_2 , \\ \dot{y}_2 &= k_3 y_2 - k_4 y_1 y_2 \end{aligned} \tag{2.59}$$

with initial values $y_1(0) = 0.4$ and $y_2(0) = 1$. Parameters to be estimated, are k_1, k_2, k_3 , and k_4 . If we insert the values $k_1 = k_2 = k_3 = 0.5$ and $k_4 = -0.2$, and try to integrate the system from $t = 0$ to $t = 10$, then every ODE solver must break down because of a singularity near $t = 3.3$.

Thus, we have to apply the shooting technique for being able to avoid singularities. If we assume that measurements for y_1 and y_2 are available, in our case obtained by simulation subject to 10 time values $t_1 = 0.1, t_2 = 0.2, \dots, t_{10} = 1$, the parameter values $p = (k_1, k_2, k_3, k_4)^T = (1, 1, 1, 1)^T$, and a subsequent perturbation of 5 %, we get the least squares problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{10} (y_1(p, t_i) - y_i^1)^2 + \sum_{i=1}^{10} (y_2(p, t_i) - y_i^2)^2 \\ p \in \mathbb{R}^4, s_1, s_2 \in \mathbb{R}^9 : \quad & y_1(p, t_i) - s_i^1 = 0, i = 1, \dots, 9, \\ & y_2(p, t_i) - s_i^2 = 0, i = 1, \dots, 9, \\ & p_l \leq p \leq p_u \end{aligned} \tag{2.60}$$

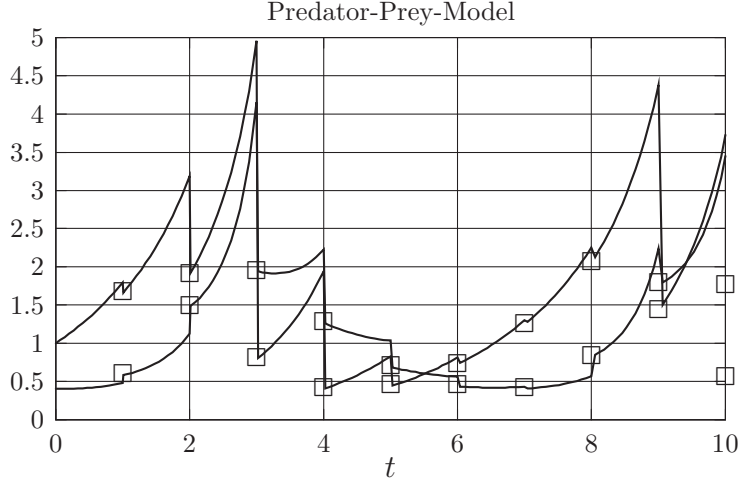


Figure 2.17: Initial Multiple Shooting Trajectories

Here we have $s_1 = (s_1^1, \dots, s_9^1)^T$ and $s_2 = (s_1^2, \dots, s_9^2)^T$. $y_1(p, t_i)$ and $y_2(p, t_i)$ are the numerical solution of the Lotka-Volterra-equation of the proceeding interval from t_{i-1} to t_i . In this case, we require that the shooting points coincide with the experimental time values, i.e., $\tau_i = t_i$, $i = 1, \dots, 9$, and $n_s = 9$, $l_t = 10$. Thus, we get a least squares problem with 22 variables and 18 additional nonlinear equality constraints.

If we start now the code *DFNLP* from $p = (0.5, 0.5, 0.5, -0.2)^T$, see above, and $s_i^k = y_i^k$, $i = 1, \dots, 9$, $k = 1, 2$, we obtain the initial trajectories displayed in Figure 2.17. The algorithm stops after 11 iterations, where the additional constraints are satisfied subject to a maximum deviation of $0.77 \cdot 10^{-7}$, see Figure 2.18. The computed optimal solution is $p = (0.984, 0.980, 1.017, 1.023)^T$.

It should be noted that the shooting method can be applied also to differential algebraic equations, see Bock, Eich, and Schlöder [53]. Additional safeguards are necessary, when restarting the integration at a shooting point to satisfy the consistency conditions.

The above example and especially Figure 2.17 show another important advantage of the shooting method. The additional artificial variables s_1, \dots, s_{n_s} require starting values for executing a data fitting algorithm, say *DFNLP*. If, however, the state variables $y_1(p, t), \dots, y_m(p, t)$ are also fitting criteria, i.e., if measurement values are available for all system variables, then these values can be used as starting values for the shooting parameters. If shooting times do not coincide with experimental times, one could compute them for example by interpolation. Thus, the shooting method leads to excellent initial trajectories we would hardly get by a trial-and-error approach. This feature explains the low number of iterations of the data fitting algorithm, we usually observe in practical applications.

The main drawback of the shooting method is that the additional variables and nonlinear equality constraints increase the complexity of the underlying data fitting problem. A typical

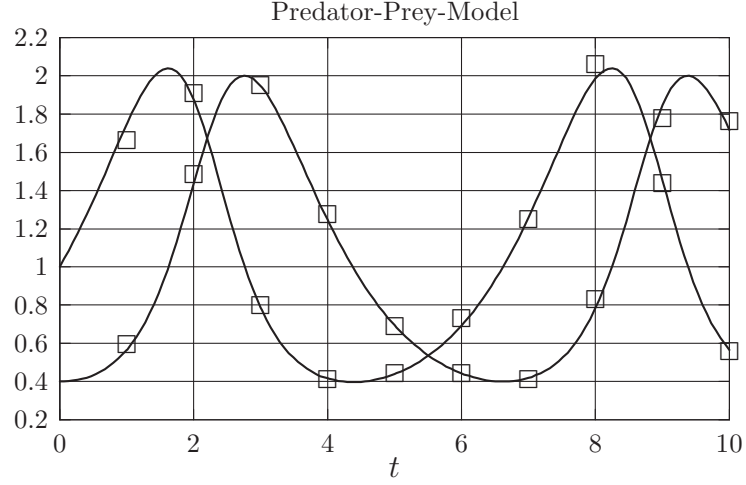


Figure 2.18: Final Multiple Shooting Trajectories

optimization algorithm handles nonlinear equality constraints by successive linearization requiring the solution of certain subproblems, where a quadratic programming or a linear least squares objective function must be minimized subject to these linear constraints, see Lindström [288].

But from (2.58) it is obvious that the Jacobian matrix of the constraints has a very special block structure that can be exploited when solving the subproblems mentioned above. If we combine the constraints (2.58) in one vector

$$g(p, s_1, \dots, s_{n_s}) = (g_1(p, s_1), \dots, g_{n_s}(p, s_{n_s}))^T$$

with

$$\begin{aligned} g_1(p, s_1) &= y^0(p, \tau_1) - s_1, \\ &\dots \\ g_{n_s}(p, s_{n_s}) &= y^{n_s-1}(p, \tau_{n_s}) - s_{n_s}, \end{aligned} \quad (2.61)$$

we get

$$\begin{aligned} \nabla_p g_1(p, s_1) &= \nabla_p y^0(p, \tau_1), \\ &\dots \\ \nabla_p g_{n_s}(p, s_{n_s}) &= \nabla_p y^{n_s-1}(p, \tau_{n_s}), \end{aligned} \quad (2.62)$$

and

$$\nabla_s g(p, s_1, \dots, s_{n_s}) = \begin{pmatrix} -I & \nabla_{s_1} y^1(p, \tau_2) & & & \\ & -I & \nabla_{s_2} y^2(p, \tau_3) & & \\ & & \ddots & \ddots & \\ & & & -I & \nabla_{s_{n_s-1}} y^{n_s-1}(p, \tau_{n_s}) \\ & & & & -I \end{pmatrix}, \quad (2.63)$$

where I represents an identity matrix of corresponding size.

Note that the evaluation of each single derivative matrix $\nabla_{s_i} y^i(p, \tau_{i+1})$ requires differentiation of the corresponding ODE subsystem subject to initial values, since each s_i is the initial value for computing $y^i(p, t)$ at $t = \tau_{i+1}$, $i = 1, \dots, n_s - 1$. However, the whole system must be integrated in any case to compute the fitting criteria. Gradients $\nabla_{s_i} y^i(p, \tau_{i+1})$ are either obtained by numerical differentiation or any other approach.

The special block structure belongs to linear equality constraints in the subproblem, so that the mn_s artificial shooting variables can be eliminated before starting the corresponding solver.

2.5.6 Boundary Value Problems

So far, we discussed only initial value problems, where first order differential equations with initial solution values at time $t = 0$ are given in the form $y(p, 0, c) = y_0(p, c)$. However, there are many applications where the solution must pass also another point, say at a final time T with solution value $y(p, T, c) = y_T(p, c)$. But the satisfaction of the additional boundary condition is only possible in the following situations:

- The underlying differential equation is a higher order system, usually of second order, where boundary values for the left and the right side are given.
- In case of a system of first order equations, there are either left or right boundary values.
- There are additional model parameters of the system to be adapted, so that a boundary value formulated as an additional nonlinear equality constraint, is to be satisfied at the optimal solution.

In the first two cases, the dynamical system can be expected to have a unique solution, so that both boundary conditions are satisfied when evaluating the fitting criterion, whereas in the second case, fulfillment of the right boundary condition is not guaranteed and can be expected at most at an optimal solution. For more details about boundary value problems (BVP) and their numerical solution see Ascher, Mattheij, and Russel [10], Mattheij and Molnaar [324], or Ascher and Petzold [11].

Let a system of second order ordinary differential equations with boundary values be given in explicit form

$$\begin{aligned} \ddot{y}_1 &= F_1(p, y, \dot{y}, t, c) \quad , \quad y_1(0) = y_1^0(p, c) \quad , \quad y_1(T) = y_1^T(p, c) \quad , \\ &\dots \\ \ddot{y}_s &= F_s(p, y, \dot{y}, t, c) \quad , \quad y_s(0) = y_s^0(p, c) \quad , \quad y_s(T) = y_s^T(p, c) \quad , \end{aligned} \tag{2.64}$$

where we omit algebraic equations for simplicity. There is no need to develop a special integration algorithm in case of a data fitting problem, since the boundary value problem is

easily transformed into a first order initial value problem with additional free parameters to be optimized, and additional state variables. From (2.64), we get the equivalent system

$$\begin{aligned}
\dot{y}_1 &= v_1 & , \quad v_1(0) &= v_1^0 , \\
&\dots & & \\
\dot{y}_m &= v_m & , \quad v_m(0) &= v_m^0 , \\
\dot{y}_1 &= F_1(p, y, v, t, c) & , \quad y_1(0) &= y_1^0(p, c) , \\
&\dots & & \\
\dot{y}_s &= F_s(p, y, v, t, c) & , \quad y_m(0) &= y_m^0(p, c) ,
\end{aligned} \tag{2.65}$$

where $v_1(t), \dots, v_m(t)$ are the additional state variables to eliminate second derivatives, and v_1^0, \dots, v_m^0 additional optimization parameters. Moreover, we have to add equality constraints of the form

$$\begin{aligned}
g_1(p) &= y_1(p, T, c) - y_1^T(p, c) = 0 , \\
&\dots & \\
g_m(p) &= y_m(p, T, c) - y_m^T(p, c) = 0
\end{aligned} \tag{2.66}$$

to the data fitting problem. Here $y(p, t, c)$ denotes the solution of (2.65) subject to p and the concentration variable c . T could be the final experimental time value, i.e., $T = l_t$. The total set of parameters to be optimized is

$$\bar{p} = (p_1, \dots, p_n, v_1^0, \dots, v_m^0)^T .$$

Example 2.15 (CARGO) *The problem is to transfer containers from a ship to a cargo truck, see Teo and Wong [502] or Elnager and Kazemi [137]. The dynamical model is described by a system of second order differential equations*

$$\begin{aligned}
\ddot{x}_1 &= u_1(t) + x_3 , \\
\ddot{x}_2 &= u_2(t) , \\
\ddot{x}_3 &= (u_1(t) + 27.0756x_3 + 2\dot{x}_2\dot{x}_3)/x_2 .
\end{aligned} \tag{2.67}$$

$u_1(t)$ and $u_2(t)$ describe the driving forces of a hoist and a trolley motor, in our case given by third order polynomials with unknown coefficients, i.e.,

$$u_k(t) = p_1^k + p_2^k t + p_3^k t^2 + p_4^k t^3 , \quad k = 1, 2 .$$

The purpose of the original formulation is to generate an optimal control test problem, where $u_1(t)$ and $u_2(t)$ are to be determined, so that a certain cost function, the swing at the end of the transfer, is to be minimized. In our case, we suppose that trajectories for the coordinates $x_1(t)$, $x_2(t)$, and $x_3(t)$ are given between two boundary points $a = (0, 22, 0)^T$ and $b = (4.22, 14.4, -0.314)^T$. The question is how to compute initial velocities $v_1^0 = \dot{x}_1(0)$,

	<i>exact solution</i>	<i>starting values</i>	<i>computed solution</i>
p_1^1	1.0	1.0	0.7955
p_2^1	-5.0	0.0	-4.0864
p_3^1	-2.0	0.0	-3.0344
p_4^1	1.0	0.0	1.3261
p_1^2	-4.0	1.0	-2.9628
p_2^2	-2.0	0.0	-1.3199
p_3^2	5.0	0.0	0.1332
p_4^2	-1.0	0.0	1.6129
v_1^0	5.0	1.0	5.0223
v_2^0	-1.0	1.0	-0.1336
v_3^0	0.0	1.0	0.0009

Table 2.7: Optimal, Start, and Computed Solution for Cargo Problem

$v_2^0 = \dot{x}_2(0)$, $v_3^0 = \dot{x}_3(0)$ and control functions $u_1(t)$, $u_2(t)$, so that the system follows the given trajectories as closely as possible at a given time $T = 2$.

Obviously, we get a boundary value problem, since we require $x_i(0) = a_i$ and $x_i(T) = b_i$ for $i = 1, 2, 3$. Since we have to expand the differential equation (2.67) by additional state variables v_1 , v_2 , v_3 to transform it into a first order system, we get three further system equations $\dot{x}_1 = v_1$, $\dot{x}_2 = v_2$, $\dot{x}_3 = v_3$. Initial values are $x_i(0) = a_i$ and the unknown ones $v_i(0) = v_i^0$ for $i = 1, 2, 3$. We try to compute these initial velocities, so that the three nonlinear equality constraints $x_i(T) - b_i = 0$ are satisfied at an optimal solution for $i = 1, 2, 3$.

Exact parameter values, starting values for DFNLP, and the computed parameters are shown in Table 2.7. DFNLP stops after 38 iterations with a maximum constraint violation of $0.29 \cdot 10^{-8}$. Figures 2.19 and 2.20 display the computed trajectories and control functions, respectively.

2.5.7 Variable Initial Times

The standard dynamical model described by ordinary differential equations or differential algebraic equations assumes that the initial time is zero, even if measurement values are not available at $t = 0$. At $t = 0$ the solution is fixed in the form $y(p, 0, c) = y_0(p, c)$. The notation indicates that initial values can be fitted by **EASY-FIT**^{ModelDesign} without applying any special techniques.

If the initial time of a real dynamical process is not zero, for example given by a $t_0 > 0$, then the model equations can be shifted easily back to zero, by replacing each occurrence of t by $t - t_0$ in case of a non-autonomous system. It is important not to forget to shift also

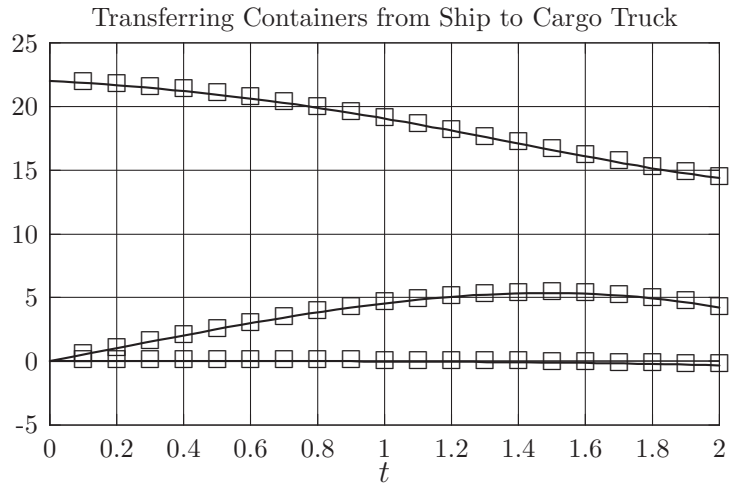


Figure 2.19: Trajectories for Cargo Problem

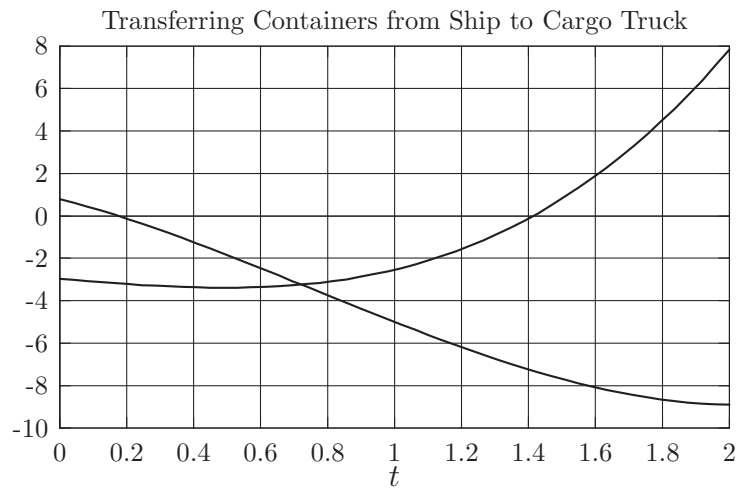


Figure 2.20: Control Functions for Cargo Problem

the measurement times t_i in the same way that is to replace all of them by $t_i - t_0$ for $i = 1, \dots, l_t$.

Another possibility to start a dynamical system at $t = t_0 > 0$, is to define zero initial values for $t = 0$, and to let the right-hand side of the differential equation become zero from $t = 0$ to $t = t_0$. At t_0 , the true initial values and differential equations are then inserted. It is necessary to declare t_0 as a switching point, see Section 2.4.3, to avoid non-continuous transitions leading eventually to numerical instabilities.

A drawback of both approaches is that the final plots are always started at $t = 0$. In the first case, shifted time values are different from the original formulation and somehow misleading, in the second case, we get zero solution values from $t = 0$ to $t = t_0$ that are out of interest for the user. The second case allows to treat t_0 as a variable initial time to be estimated, if we declare it as an optimization parameter.

However, we are supposing that in most practical situations, the initial time is indeed zero. To become a bit more flexible in case of non-zero initial times, **EASY-FIT**^{ModelDesign} allows to define negative experimental time values. If the first measurement time t_1 is negative, then the integration is started at $t = t_1 < 0$ with initial value $y(p, t_1, c) = y_0(p, c)$, which may depend also on the parameters to be estimated, and the concentration parameter. A possible application is found in the example.

Example 2.16 (PHA_DYN1/2/3) *We consider a pharmacodynamic process of the form*

$$\begin{aligned}\dot{x}_1 &= k^p x_2, \\ \dot{x}_2 &= k_1^p (x_0 - x_2 - x_3)(s_0 - x_2) - (k_1^m + k^p) x_2, \\ \dot{x}_3 &= k_2^p (x_0 - x_2 - x_3)(c - x_3) - k_2^m x_3\end{aligned}\tag{2.68}$$

with constants $s_0 = 1$, $x_0 = 0.5$, and $k_1^m = 10$, and c is a concentration parameter. k^p , k_1^p , k_2^p , and k_2^m are parameters to be estimated, with initial values $k^p = 2$, $k_1^p = 5$, $k_2^p = 3$, and $k_2^m = 10$. Experimental data are shown in Table 2.8 for five different concentration values $c_1 = 0.005$, $c_1 = 0.05$, $c_2 = 0.25$, $c_3 = 0.5$, $c_4 = 0.75$, and $c_5 = 1$.

First, we observe that experimental data are only available for x_1 , and that we got positive values at $t = 0$. In other words, each of the five separate processes starts at an unknown initial time $\tau_j < 0$, $j = 1, \dots, 5$. The first attempt could be to define additional optimization parameters for all initial values, all together 15 additional optimization parameters. The number of iterations n_{it} , the final residual value, and the optimal parameter set are listed in Table 2.9, case 1. However, the results are incorrect since we do not take into account that initial times for the three equations (2.68) must coincide for each concentration value. Thus, we suppose that there is one fixed initial time $t_0 = -0.1$. The results are also shown in Table 2.9, case 2. Finally we introduce one variable initial time for each of the 5 concentrations, and start the data fitting run at $\tau_j = -0.1$, $j = 1, \dots, 5$. The integration is initialized at $t = -0.2$. Initial values at -0.2 and right-hand side of corresponding equations are set to zero, see also Figures 2.21 to 2.23 for plots of the state variables over time and concentration. We know that initial values of the process under consideration are zero.

t_i	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}
0	0.014	0.016	0.015	0.009	0.007
0.1666	0.065	0.059	0.051	0.038	0.031
0.3333	0.117	0.100	0.088	0.069	0.055
0.5	0.167	0.142	0.123	0.099	0.080
0.6666	0.214	0.181	0.157	0.129	0.112
0.8333	0.264	0.220	0.193	0.159	0.137
1	0.311	0.261	0.228	0.193	0.166

Table 2.8: Experimental Data

$case$	n_{it}	$residual$	k^p	k_1^p	k_2^p	k_2^m
1	30	0.000063	3.167	3.403	10.804	7.041
2	8	0.000375	3.382	3.320	13.667	6.709
3	6	0.000125	2.951	3.763	12.210	7.008

Table 2.9: Performance Results and Computed Solution

Case 3 of Table 2.9 contains achieved performance results and computed parameters. The best residual is obtained for case 1. But the model is incorrect, as in case 2. For case 3, we get the most appropriate results.

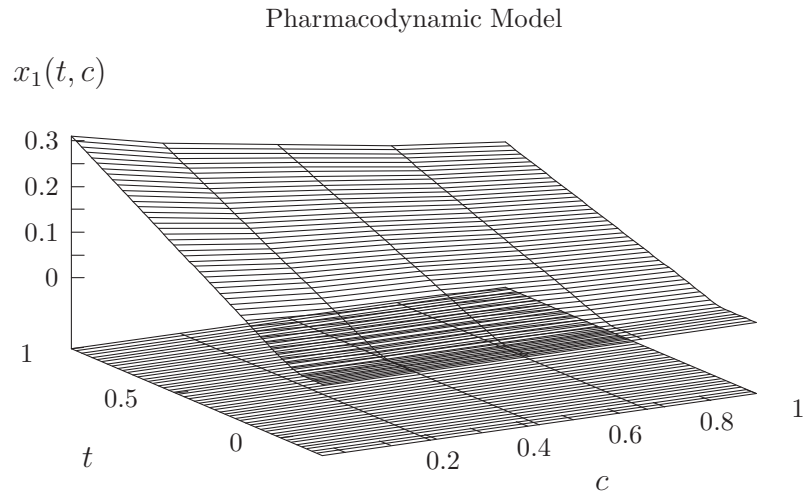


Figure 2.21: State Variable $x_1(t, c)$ over Time t and Concentration c

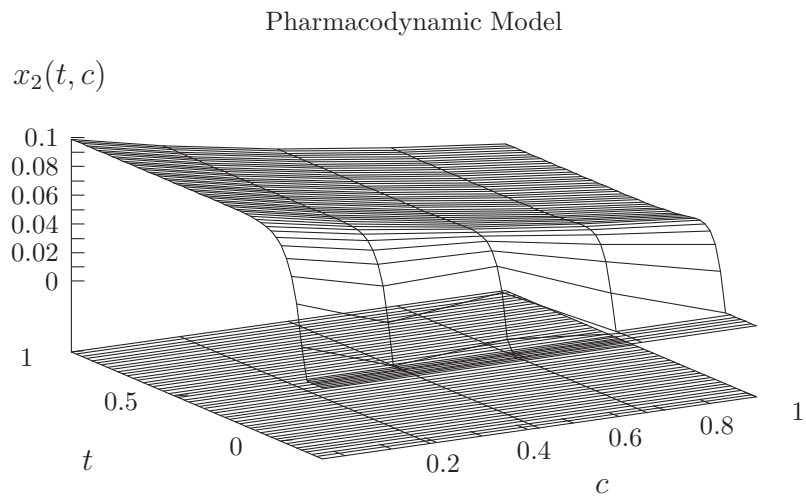


Figure 2.22: State Variable $x_2(t, c)$ over Time t and Concentration c

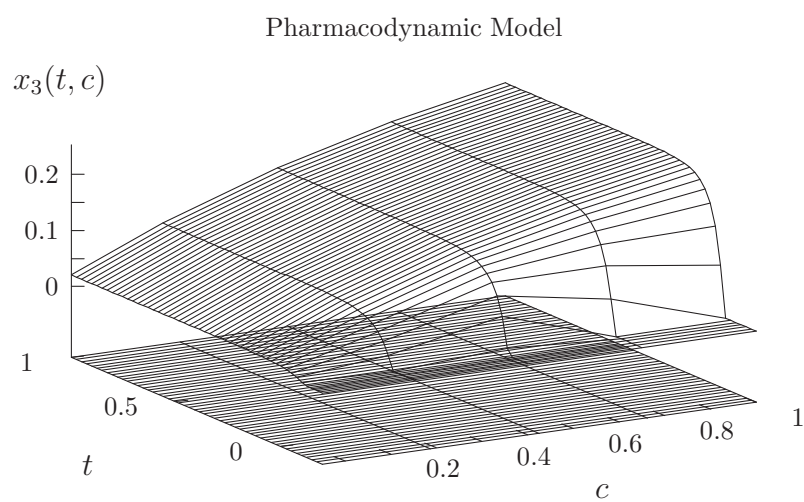


Figure 2.23: State Variable $x_3(t, c)$ over Time t and Concentration c

2.6 Partial Differential Equations

2.6.1 Standard Formulation

Now we proceed from r data sets

$$(t_i, y_i^k), \quad i = 1, \dots, l_t, \quad k = 1, \dots, r, \quad (2.69)$$

where l_t time values and $l_t r$ corresponding measurement values are defined. To simplify the analysis, we omit the additional independent model variable c called *concentration* in the previous sections.

In its most simple form a system of time-dependent one-dimensional partial differential equations is given by

$$u_t = F(p, u, u_x, u_{xx}, x, t) . \quad (2.70)$$

The expanded form is

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= F_1(p, u, u_x, u_{xx}, x, t) , \\ &\dots \\ \frac{\partial u_{n_p}}{\partial t} &= F_{n_p}(p, u, u_x, u_{xx}, x, t) , \end{aligned} \quad (2.71)$$

if we consider the individual coefficients of F and u , i.e., if

$$F(p, u, u_x, u_{xx}, x, t) = (F_1(p, u, u_x, u_{xx}, x, t), \dots, F_{n_p}(p, u, u_x, u_{xx}, x, t))^T$$

and $u = (u_1, \dots, u_{n_p})^T$, respectively. We denote the solution of (2.70) by $u(p, x, t)$, since it depends on the time value t , the space value x , and the actual parameter value p .

Also initial and boundary conditions may depend on the parameter vector to be estimated. Since the starting time is assumed to be zero, initial conditions have the form

$$u(p, x, 0) = u_0(p, x) \quad (2.72)$$

and are defined for all $x \in [x_L, x_R]$. For both end points x_L and x_R we allow Dirichlet or Neumann boundary values

$$\begin{aligned} u(p, x_L, t) &= u^L(p, t) , \\ u(p, x_R, t) &= u^R(p, t) , \\ u_x(p, x_L, t) &= \hat{u}^L(p, t) , \\ u_x(p, x_R, t) &= \hat{u}^R(p, t) \end{aligned} \quad (2.73)$$

for $0 < t \leq T$, where T is the final integration time, for example the last experimental time value t_{l_t} . The availability of all boundary functions is of course not required. Their particular choice depends on the structure of the PDE model, for example whether second partial derivatives exist in the right-hand side or not.

To indicate that the fitting criteria $h_k(p, t)$ depend also on the solution of the dynamical equation at the corresponding fitting point and its derivatives, where k denotes the index of a measurement set, we use the notation

$$h_k(p, t) = \bar{h}_k(p, u(p, x_k, t), u_x(p, x_k, t), u_{xx}(p, x_k, t), t) \quad . \quad (2.74)$$

Each set of experimental data is assigned a spatial variable value $x_k \in [x_L, x_R]$, $k = 1, \dots, r$, where r denotes the total number of measurement sets. Some or all of the x_k -values may coincide, if different measurement sets are available at the same local position. Since partial differential equations are discretized by the method of lines, the fitting points x_k are rounded to the nearest line.

Also in this case, we assume that the observation times are strictly increasing, and get the objective functions

$$\sum_{k=1}^r \sum_{i=1}^{l_t} (w_i^k (h_k(p, t_i) - y_i^k))^2 \quad (2.75)$$

for the least squares norm,

$$\sum_{k=1}^r \sum_{i=1}^{l_t} w_i^k |h_k(p, t_i) - y_i^k| \quad (2.76)$$

for the L_1 -norm, and

$$\max_{k=1, \dots, r; i=1, \dots, l_t} w_i^k |h_k(p, t_i) - y_i^k| \quad (2.77)$$

for the maximum-norm.

Example 2.17 (HEAT_A) *To illustrate the standard formulation, we consider a very simple parabolic PDE, the heat equation*

$$u_t = p_1 u_{xx} \quad (2.78)$$

with a diffusion coefficient $p_1 > 0$. The spatial variable x varies from 0 to 1, and the time variable is non-negative, i.e., $t \geq 0$. The initial heat distribution for $t = 0$ is

$$u(p, x, 0) = p_2 \sin(\pi x) \quad (2.79)$$

for all $x \in (0, 1)$, and Dirichlet boundary values

$$u(p, 0, t) = u(p, 1, t) = 0 \quad (2.80)$$

for all $t \geq 0$ are set. It is easy to verify by insertion that

$$u(p, x, t) = p_2 \exp(-p_1 \pi^2 t) \sin(\pi x)$$

is the exact solution in case of $p = (p_1, p_2)^T = (1, 1)^T$.

Heat Equation

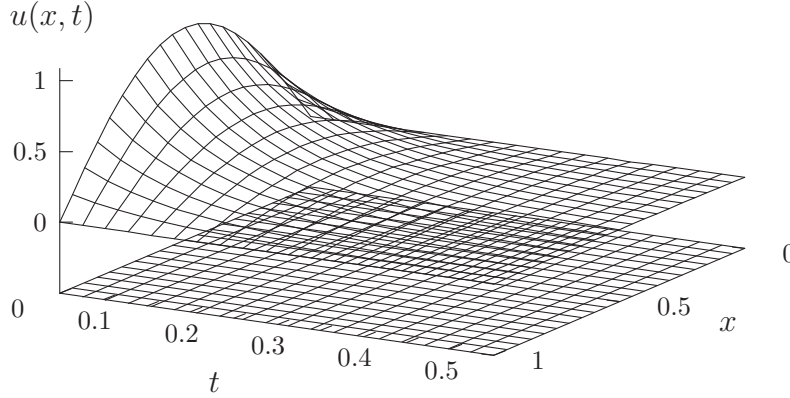


Figure 2.24: Solution of a Parabolic Equation

To construct a data fitting problem, we simulate experimental data for $p = (1, 1)^T$ at 9 time values $t_1 = 0.1, \dots, t_9 = 0.9$ and 3 spatial values $x_1 = 0.25, x_2 = 0.5$, and $x_3 = 0.75$. Subsequently a uniformly distributed error of 5 % is added. Thus, the data fitting problem consists of minimizing the function

$$\sum_{k=1}^3 \sum_{i=1}^9 (u(p, x_k, t_i) - y_i^k)^2$$

over all $p \in \mathbb{R}^2$.

The partial differential equation is discretized by 31 lines and a fifth order difference formula. The resulting system of 29 ordinary differential equations is solved by an implicit Runge-Kutta method with integration accuracy 10^{-6} . When starting the least squares algorithm DFNLP of Schittkowski [429] with termination accuracy 10^{-10} from $p_0 = (2, 2)^T$, we get the solution $p^* = (0.98, 0.97)^T$ after 9 iterations. The final surface plot of the state variable $u(x, t)$ is shown in Figure 2.24.

2.6.2 Partial Differential Algebraic Equations

One-dimensional partial differential algebraic equations (PDAE) are based on the same model structure as one-dimensional, time-dependent partial differential equations. The only difference is that additional algebraic equations are permitted as in case of DAE's. Typical examples are higher order partial differential equations, for example

$$u_t = f(p, u, u_{xxxx}, x, t) \quad ,$$

or distributed systems of the form

$$\begin{aligned} u_t &= f(p, u, v, x, t) , \\ v_x &= g(p, u, v, x, t) \end{aligned}$$

with initial values $u(p, x, 0) = u(p, x)$, $v(p, 0, t) = v(p, t)$.

We proceed from the general explicit formulation

$$\begin{aligned} \frac{\partial u_d}{\partial t} &= F_d(p, u, u_x, u_{xx}, x, t) , \\ 0 &= F_a(p, u, u_x, u_{xx}, x, t) , \end{aligned} \tag{2.81}$$

where $x \in \mathbb{R}$ is the spatial variable with $x_L \leq x \leq x_R$, and $0 < t \leq T$. Initial and boundary conditions are the same as in the previous section, see (2.72) and (2.73).

But now the state variables are divided into n_d so-called differential variables $u_d = (u_1, \dots, u_{n_d})^T$ and n_a algebraic variables $u_a = (u_{n_d+1}, \dots, u_{n_d+n_a})^T$, where the number of algebraic variables is identical to the number of algebraic equations summarized by the vector F_a . The dynamical system (2.81) is also written in the equivalent form

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= F_1(p, u, u_x, u_{xx}, x, t) , \\ &\dots \\ \frac{\partial u_{n_d}}{\partial t} &= F_{n_d}(p, u, u_x, u_{xx}, x, t) , \\ 0 &= F_{n_d+1}(p, u, u_x, u_{xx}, x, t) , \\ &\dots \\ 0 &= F_{n_d+n_a}(p, u, u_x, u_{xx}, x, t) , \end{aligned} \tag{2.82}$$

if we consider the individual coefficient functions $F = (F_1, \dots, F_{n_d+n_a})^T$.

However, we must treat initial and boundary conditions with more care. We have to guarantee that at least existing boundary conditions satisfy the algebraic equations, for example

$$\begin{aligned} 0 &= F_a(p, u(p, x_L, t), u_x(p, x_L, t), u_{xx}(p, x_L, t), x_L, t) , \\ 0 &= F_a(p, u(p, x_R, t), u_x(p, x_R, t), u_{xx}(p, x_R, t), x_R, t) , \end{aligned} \tag{2.83}$$

where u is the combined vector of all differential and algebraic state variables. If initial conditions for discretized algebraic equations are violated, i.e., if equation

$$0 = F_a(p, u(p, x, 0), u_x(p, x, 0), u_{xx}(p, x, 0), x, 0) \tag{2.84}$$

is violated after inserting Dirichlet or Neumann boundary values and corresponding approximations for spatial derivatives, then the corresponding system of nonlinear equations must be solved numerically proceeding from given initial values. In other words, consistent initial

values can be computed automatically, where the given data serve as starting parameters for the nonlinear programming algorithm applied.

But even if we succeed to find consistent initial values for (2.84) *by hand*, we have to take into account that the algebraic state variables and the spatial derivatives in the dynamical equation (2.84) are approximated numerically by the method of lines and suitable difference or any similar formulae. The corresponding discretized equations of the DAE system are in general not consistent, or, more precisely, are satisfied only within the given discretization accuracy. Thus, we have to assume that the resulting DAE system is an index-1-system unless it is guaranteed that consistent initial values for the discretized DAE are available, see for example Caracotsios and Stewart [76] for a similar approach.

Example 2.18 (HEAT_A) *We consider again Example 2.17 now formulated as a first order PDAE*

$$\begin{aligned} u_t &= Dv_x , \\ 0 &= v - u_x \end{aligned} \tag{2.85}$$

with diffusion coefficient D . The spatial variable x varies from 0 to 1, and the time variable is non-negative, i.e., $t \geq 0$. The initial heat distribution for $t = 0$ is

$$\begin{aligned} u(p, x, 0) &= a \sin(\pi x) , \\ v(p, x, 0) &= 0 \end{aligned} \tag{2.86}$$

for all $x \in (0, 1)$. Dirichlet boundary values are the same as before, see (2.80), and the parameter vector $p = (D, a)^T$ is to be estimated subject to the same simulated experimental data computed for Example 2.17.

When starting the same least squares and integration algorithms as before, we get an identical solution after 5 iterations. The initial values are not consistent, but are easily computed within machine accuracy in two iterations by the nonlinear programming code NLPQLP of Schittkowski [427, 440, 449]. Stopping tolerance is set to 10^{-10} . The maximum error of the algebraic equation along the lines $x_1 = 0.25$, $x_2 = 0.5$ and $x_3 = 0.75$ is $0.11 \cdot 10^{-8}$. The corresponding plot for the algebraic variable $v(p, x, t)$ is shown in Figure 2.25.

If, on the other hand, the initial values for v are changed to

$$\begin{aligned} u(p, x, 0) &= a \sin(\pi x) , \\ v(p, x, 0) &= \pi a \cos(\pi x) , \end{aligned}$$

we get theoretically consistent initial values. However, spatial derivatives are approximated numerically, so that we have to relax the termination tolerance for executing NLPQLP to 10^{-7} according to the discretization accuracy, to avoid re-calculation of consistent initial values of the discretized DAE.

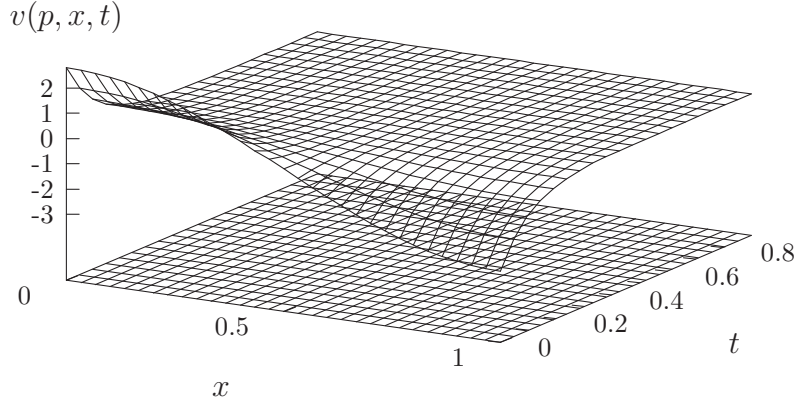


Figure 2.25: Surface Plot of Algebraic State Variable

2.6.3 Flux Functions

Again, we proceed from a system of n_d differential and n_a algebraic equations in explicit formulation (2.81), where the state variables consist of n_d differential variables u_d and n_a algebraic variables u_a , i.e., $u = (u_d, u_a)^T$. But now we introduce an additional *flux* function $f(p, u, u_x, x, t)$, i.e., we suppose that our dynamical system is given by

$$\begin{aligned} \frac{\partial u_d}{\partial t} &= F_d(p, f(p, u, u_x, x, t), f_x(p, u, u_x, x, t), u, u_x, u_{xx}, x, t), \\ 0 &= F_a(p, f(p, u, u_x, x, t), f_x(p, u, u_x, x, t), u, u_x, u_{xx}, x, t), \end{aligned} \quad (2.87)$$

where $x \in \mathbb{R}$ is the spatial variable with $x_L \leq x \leq x_R$, and $0 < t \leq T$. Initial and boundary conditions are the same as before, see (2.72) and (2.73).

Flux functions are useful in two situations. First they facilitate the declaration of highly complex model functions given by their flux formulations. In these cases, it is often difficult or impossible to get the spatial derivatives in analytical form, and one has to apply a first order discretization scheme to the entire flux function.

Example 2.19 (MOL_DIFF) *The model describes the diffusion of molecules, where a flux function is given by*

$$f(p, c, c_x, x, t) = D \exp(\beta(c - 1))c_x$$

and the diffusion equation is

$$c_t = f_x(p, c, c_x, x, t) = \frac{\partial}{\partial x} \left(D \exp(\beta(c - 1))c_x \right).$$

k	x_k	y_1^k
1	-218	0.8865
2	-182	0.8737
3	-145	0.8609
4	-109	0.8511
5	-72	0.8063
6	-36	0.6891
7	0	0.6678
8	36	0.6227
9	72	0.6366
10	109	0.6530

Table 2.10: Experimental Values

In this simple situation, the flux function is avoided easily by analytical differentiation by hand, i.e., the dynamical equation is equivalent to

$$c_t = D \exp(\beta(c-1))c_{xx} + D\beta \exp(\beta(c-1))c_x^2 .$$

Initial value is $c(p, x, 0) = 0$, if $x < 0$, and $c(p, x, 0) = 1$ otherwise. Dirichlet boundary values are $c(p, -500, t) = 1$ and $c(p, 500, t) = 0$. The remaining coefficients are supposed to be parameters to be estimated, i.e., $p = (D, \beta)^T$. Experimental data are available for one time value $t_1 = 353$ at 10 different spatial values, see Table 2.10.

When applying a fifth-order difference formula with 21 lines, DFNLP computes the solution $D = 627.0$, $\beta = 3.608$ within 26 iterations starting from $D = 400$ and $\beta = 3$. Maximum deviation of measurement values from experimental data is 10.26 %. The resulting surface plot is shown in Figure 2.26.

Another reason for using flux functions is to apply special upwind formulae in case of hyperbolic equations, when usual approximation schemes break down. Typical reason is the propagation of shocks over the integration interval, enforced by non-continuous initial and boundary conditions. In most situations, advection or transport equations are considered,

$$u_t + f_x(p, u) = g(p, u, u_x, u_{xx}, x, t) , \quad (2.88)$$

where

$$F(p, f, f_x, u, u_x, u_{xx}, x, t) = g(p, u, u_x, u_{xx}, x, t) - f_x(p, u, u_x, x, t) .$$

Example 2.20 (BURGER_E) *We consider a very famous example now, the so-called viscous Burgers' equation defined by the flux function*

$$f(u) = 0.5u^2$$

Molecule Diffusion

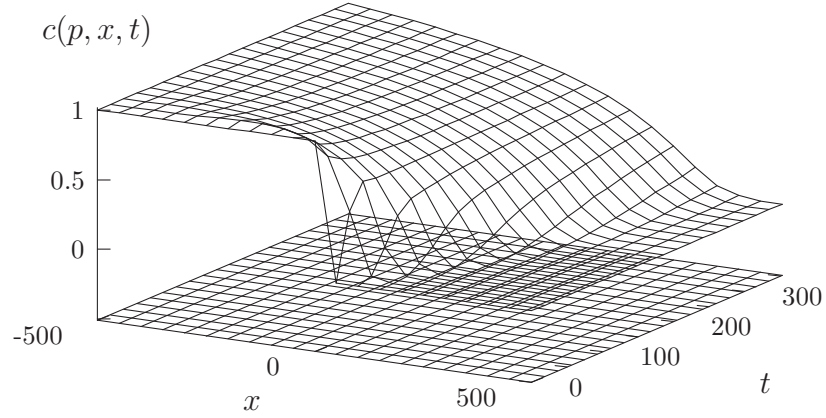


Figure 2.26: Diffusion Variable Defined by Flux Function

and

$$u_t + f_x(u) = pu_{xx} \quad ,$$

see Thomas [503]. In this case, the exact solution is known,

$$u(p, x, t) = \frac{1}{1 + \exp(x/(2p) - t/(4p))}$$

and can be used to simulate 54 measurement data at $t = 0.1, \dots, t = 0.9$ and $x = 0.1, \dots, x = 0.6$ subject to $p = 0.01$. Subsequently, these data are perturbed with an error of 5 %. The exact solution is used to generate initial values and Dirichlet boundary conditions. The partial differential equation is discretized by 51 lines and a second order upwind scheme. The least squares code DFNLP is started at $p_0 = 1$ together with an implicit solver for the system of 49 ordinary differential equations. After 39 iterations we get the estimated parameter $p = 0.01031$. The resulting surface plot is shown in Figure 2.27. We see that the viscous, i.e., the parabolic part of the equation smoothes the edges.

2.6.4 Coupled Ordinary Differential Algebraic Equations

A particular advantage of applying the method of lines for discretizing a partial differential equation is the possibility to couple additional ordinary differential algebraic equations to

Burgers' Equation

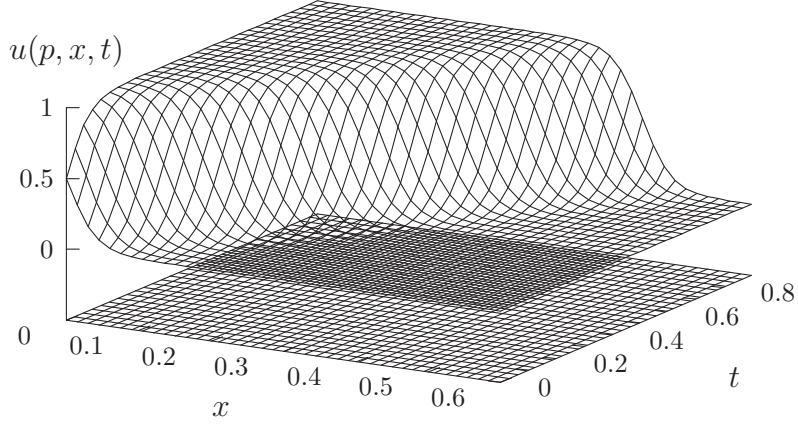


Figure 2.27: Solution of a Hyperbolic Equation

the given partial ones. We proceed from the general explicit formulation

$$\begin{aligned} \frac{\partial u_d}{\partial t} &= F_d(p, u, u_x, u_{xx}, v, x, t) , \\ 0 &= F_a(p, u, u_x, u_{xx}, v, x, t) , \end{aligned} \quad (2.89)$$

confer (2.81), where $x \in \mathbb{R}$ is the spatial variable with $x_L \leq x \leq x_R$, and $0 < t \leq T$. Initial and boundary conditions are the same as in the previous section, see (2.72) and (2.73),

$$u(p, x, 0) = u_0(p, x) \quad (2.90)$$

to be satisfied for all $x \in [x_L, x_R]$, and

$$\begin{aligned} u(p, x_L, t) &= u^L(p, v, t) , \\ u(p, x_R, t) &= u^R(p, v, t) , \\ u_x(p, x_L, t) &= \hat{u}^L(p, v, t) , \\ u_x(p, x_R, t) &= \hat{u}^R(p, v, t) \end{aligned} \quad (2.91)$$

for $0 < t \leq T$, where either Dirichlet or Neumann or any mixed boundary conditions must be defined. Also these boundary conditions may depend on the coupled differential and algebraic variables, for example when boundary conditions are given in form of ordinary differential equations or in implicit form.

The right-hand side of the partial differential equation depends in addition on the solution of a system of ordinary differential algebraic equations $v = (v_d, v_a)^T \in \mathbb{R}^{n_c}$, given in the form

$$\begin{aligned}
\frac{\partial v_1}{\partial t} &= G_1(p, u(p, x_1, t), u_x(p, x_1, t), u_{xx}(p, x_1, t), v, t) , \\
&\dots \\
\frac{\partial v_{n_{dc}}}{\partial t} &= G_{n_{dc}}(p, u(p, x_{n_{dc}}, t), u_x(p, x_{n_{dc}}, t), u_{xx}(p, x_{n_{dc}}, t), v, t) , \\
0 &= G_{n_{dc}+1}(p, u(p, x_{n_{dc}+1}, t), u_x(p, x_{n_{dc}+1}, t), u_{xx}(p, x_{n_{dc}+1}, t), v, t) , \\
&\dots \\
0 &= G_{n_c}(p, u(p, x_{n_c}, t), u_x(p, x_{n_c}, t), u_{xx}(p, x_{n_c}, t), v, t)
\end{aligned} \tag{2.92}$$

for $j = 1, \dots, n_c$, where $u(p, x, t)$ is the solution vector of the partial differential algebraic equation. Here x_j are any x -coordinate values where the corresponding ordinary differential algebraic equation is to be coupled to the partial one. Some of these values may coincide. When discretizing the system by the method of lines, they have to be rounded to the nearest neighboring line.

Also initial values

$$v(p, 0) = v_0(p) \tag{2.93}$$

may depend again on the parameters to be estimated. A solution of the coupled system depends on the spatial variable x , the time variable t , the parameter vector p , and is therefore written in the form $v(p, t)$ and $u(p, x, t)$.

To indicate that also the fitting criteria $h_k(p, t)$ depend on the solution of the differential equation at the corresponding fitting point and its first and second spatial derivatives, we use the notation

$$h_k(p, t) = \bar{h}_k(p, u(p, x_k, t), u_x(p, x_k, t), u_{xx}(p, x_k, t), v(p, t), t) , \tag{2.94}$$

see also (2.74). k denotes the index of a measurement set. Also fitting points x_k are rounded to their nearest line when discretizing the system.

Coupled ordinary differential equations can be used to define a fitting criterion, for example if the flux into or out of a system is investigated. Another reason is that they allow to replace Dirichlet or Neumann boundary conditions by differential equations, see the subsequent example.

Example 2.21 (SALINE) *The model describes the diffusion of drug in a saline solution through membrane, see Spoelstra and van Wyk [489]. There are two differential state variables c and z with corresponding system equations*

$$c_t = \frac{\partial}{\partial x} (a(c, z)c_x) - qc$$

and

$$z_t = qc ,$$

	p^*	p_0	p
D	0.1	1.0	0.103
b	0.3	1.0	0.184
q	1.0	0.5	1.420
c_0	5.0	1.0	5.063

Table 2.11: Exact, Starting, and Computed Parameter Values

no algebraic equations, and two ordinary differential equations coupled at boundary points $x_L = 0$ and $x_R = 0.1$,

$$\begin{aligned}\dot{c}_L &= Da(c(p, x_L, t), z(p, x_L, t)) c_x(p, x_L, t) , \\ \dot{c}_R &= -Da(c(p, x_R, t), z(p, x_R, t)) c_x(p, x_R, t) .\end{aligned}$$

Here we have

$$a(c, z) = \exp(b(z/(1+c))^2) .$$

Initial values are $c(p, x, 0) = 0$, $z(p, x, 0) = 0$, $c_L(p, 0) = 0$, and $c_R(p, 0) = c_0$. The coupled ordinary differential equations are inserted to model the flux into and out of the system. Thus, the boundary values are $c(p, x_L, t) = c_L(p, t)$ and $c(p, x_R, t) = c_R(p, t)$, and the fitting criteria are the two functions

$$h(p, t) = k(c_L(p, t), c_R(p, t))^T$$

with a scaling constant $k = 100,000$. The remaining coefficients are supposed to be parameters to be estimated, i.e., $p = (D, b, q, c_0)^T$.

Experimental data are simulated at 10 equidistant time values between 0 and 5 subject to a 5 % error. Exact, starting, and final parameter values are given in Table 2.11. They are obtained by executing DFNLP with a termination tolerance of 10^{-7} terminating after 56 iterations. The partial differential equation is discretized at 21 lines and a three-point difference formula. The resulting system of ODE's is solved by an implicit method with an error tolerance 10^{-5} . Function and data plot is found in Figure 2.28 and the state variable c is displayed in Figure 2.29.

If coupled algebraic equations violate initial conditions (2.93) after a suitable discretization of the partial derivatives u_x and u_{xx} , Newton's method can be applied to compute consistent initial values. The equations are added to the algebraic partial differential equations and the whole system of nonlinear equations must be solved simultaneously.

Algebraic differential equations are highly useful in case of implicit boundary conditions, since the spatial coupling values may coincide with boundary points. Each algebraic equation requires also the declaration of a corresponding algebraic variable, for instance for the partial state variable or its derivative at a boundary point. Thus, one has to define also some trivial Dirichlet or Neumann conditions containing only the algebraic variable at the right-hand side.

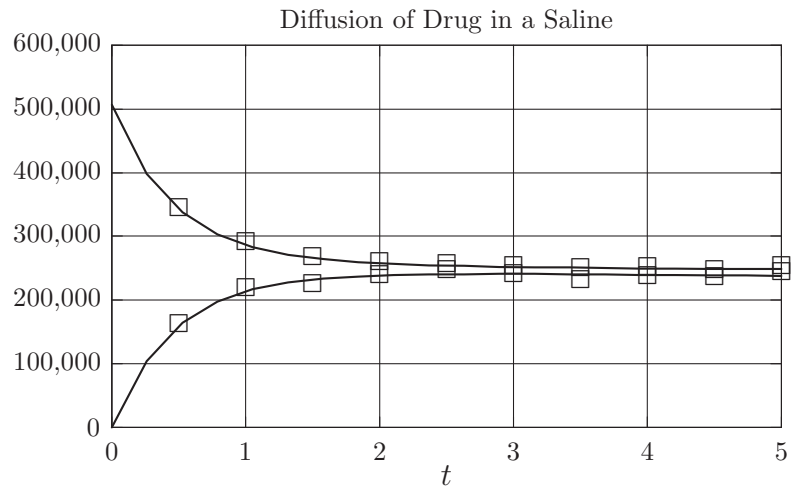


Figure 2.28: Function and Data Plot

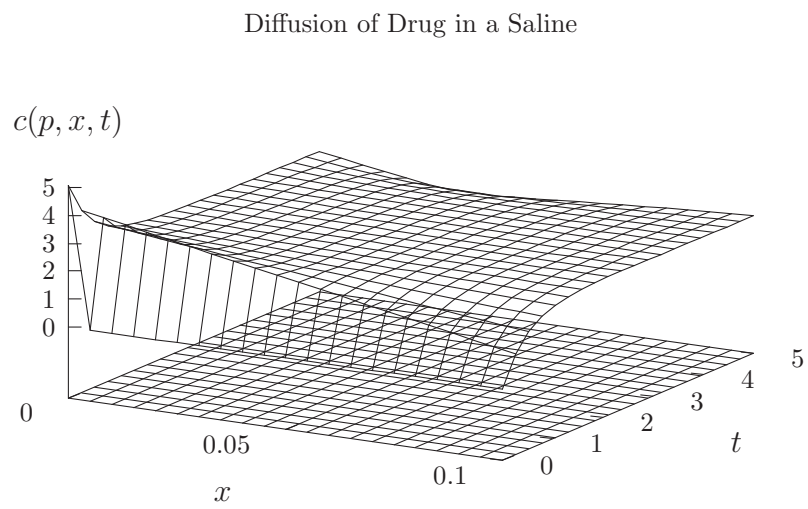


Figure 2.29: State Variable

Example 2.22 (HEAT_NLB) We consider a simple parabolic equation, see also Tröltzsch [510],

$$y_t = y_{xx}$$

with initial value $y(x, 0) = \cos(x)$, a homogeneous left Neumann boundary condition $y(0, t) = 0$, and another implicitly defined boundary condition

$$u(t) + e(t) - y(x_R, t)^4 - y_x(x_R, t) = 0$$

at the right end point, $x_R = 0.785398$. We have

$$e(t) = 0.25 \exp(-4t) - \frac{\exp(t) - \exp(1/3)}{\exp(2/3) - \exp(1/3)}$$

and a ramp function

$$u(t) = \begin{cases} 0, & \text{if } t \leq t_1, \\ \frac{t - t_1}{t_2 - t_1}, & \text{if } t_1 < t \leq t_2, \\ 1, & \text{if } t > t_2, \end{cases}$$

to control the system at the right boundary. t_1 and t_2 are given and we measure the distance of $y(x_R, t)$ from $\exp(-t) \cos(x)$ at 10 equidistant grid points within 0 and 1. The first possibility is to define a Neumann boundary condition at the right end point x_R in the form

$$y_x(x_R, t) = u(t) + e(t) - y(x_R, t)^4.$$

Alternatively we have the possibility to treat the implicit boundary condition as a coupled algebraic equation

$$u(t) + e(t) - y(x_R, t)^4 - v(t) = 0$$

with an algebraic variable $v(t)$. Together with the trivial Neumann condition $y_x(x_R, t) = v(t)$ we get an equivalent formulation. A third possibility is to consider the algebraic equation

$$u(t) + e(t) - v^4 - y_x(x_R, t) = 0,$$

but now formulated for the Dirichlet boundary condition $y(x_R, t) = v(t)$.

Since the first two options are more or less identical, we perform some numerical tests with the original formulation with a Neumann boundary condition (case A) and the coupled algebraic equation for getting a Dirichlet boundary condition (case B). A five-point difference formula is used for approximating first and second derivatives. The discretized system of ordinary differential equations (case A) or differential algebraic equations (case B) is solved with error tolerance 10^{-7} . A few simulations are performed for $t_1 = 0.3$ and $t_2 = 0.7$ with increasing number of lines n , see Table 2.12, where the computed residual is listed. There are no differences within integration accuracy under 11 or more lines.

n	case A	case B
7	0.752, 347, 16	0.752, 591, 03
11	0.752, 634, 19	0.752, 634, 20
21	0.752, 633, 46	0.752, 633, 47

Table 2.12: Neumann Boundary Condition Versus Coupled Algebraic Equation

2.6.5 Integration Areas and Transition Conditions

Now we extend the model structure to take different integration intervals into account. Possible reasons are the diffusion of a substrate through different media, for example, where we want to describe the transition from one area to the next by special conditions. Since these transition conditions may become non-continuous, we need a more general formulation and have to adapt the discretization procedure.

The general model is defined by a system of n_d one-dimensional partial differential equations and n_a algebraic equations in one or more spatial intervals, see also Schittkowski [433]. These intervals are given by the outer boundary values x_L and x_R that define the total integration interval with respect to the space variable x , and optionally some additional internal transition points $x_1^a, \dots, x_{m_a-1}^a$. Thus, we get a sequence of $m_a + 1$ boundary and transition points

$$x_0^a = x_L < x_1^a < \dots < x_{m_a-1}^a < x_{m_a}^a = x_R . \quad (2.95)$$

For each integration interval, we define a system of partial differential equations of the form

$$\begin{aligned} \frac{\partial u_d^i}{\partial t} &= F_d^i(p, f^i(p, u^i, u_x^i, x, t), f_x^i(p, u^i, u_x^i, x, t), u^i, u_x^i, u_{xx}^i, v^i, x, t) , \\ 0 &= F_a^i(p, f^i(p, u^i, u_x^i, x, t), f_x^i(p, u^i, u_x^i, x, t), u^i, u_x^i, u_{xx}^i, v^i, x, t) , \end{aligned} \quad (2.96)$$

where $x \in \mathbb{R}$ is the spatial variable with $x_{i-1}^a < x < x_i^a$ for $i = 1, \dots, m_a$, $t \in \mathbb{R}$ the time variable with $0 < t \leq T$, $v \in \mathbb{R}^{n_c}$ the state variable of the coupled system of ordinary differential algebraic equations, $u^i = (u_d^i, u_a^i)^T \in \mathbb{R}^{n_p}$ the state variables consisting of the partial differential variables $u_d^i \in \mathbb{R}^{n_d}$ and partial algebraic variables $u_a^i \in \mathbb{R}^{n_a}$, and $p \in \mathbb{R}^n$ is the parameter vector to be identified by the data fitting algorithm. See also (2.89) for coupled ordinary differential equations, (2.87) for flux functions, (2.82) for algebraic equations, and (2.70) for the most simple standard formulation.

Optionally, the right-hand side may depend also on a so-called flux function $f^i(p, u^i, u_x^i, x, t)$, where we omit for simplicity a possible dependency from coupled ordinary differential equations. A solution depends on the spatial variable x , the time variable t , the parameter vector p , the corresponding integration interval, and is therefore written in the form $v(p, t)$ and $u^i(p, x, t)$ for $i = 1, \dots, m_a$.

For both boundary points x_L and x_R we allow Dirichlet or Neumann conditions of the

form

$$\begin{aligned}
u^1(p, x_L, t) &= u^L(p, v, t) \\
u^{n_a}(p, x_R, t) &= u^R(p, v, t) \\
u_x^1(p, x_L, t) &= \hat{u}^L(p, v, t) \\
u_x^{n_a}(p, x_R, t) &= \hat{u}^R(p, v, t)
\end{aligned} \tag{2.97}$$

for $0 < t \leq t_s$. Again we do not require the evaluation of all boundary functions. Instead a user may omit some of them depending on the structure of the dynamical model. Note that boundary information is also contained in coupled ordinary differential algebraic equations.

Transition conditions between the different areas may be defined in addition. They are allowed at most at transition points and have the form

$$\begin{aligned}
u^i(p, x_i^a, t) &= c_i^R(p, u^{i+1}(p, x_i^a, t), v, t) \ , \\
u^{i+1}(p, x_i^a, t) &= c_i^L(p, u^i(p, x_i^a, t), v, t) \ , \\
u_x^i(p, x_i^a, t) &= \hat{c}_i^R(p, u^{i+1}(p, x_i^a, t), u_x^{i+1}(p, x_i^a, t), v, t) \ , \\
u_x^{i+1}(p, x_i^a, t) &= \hat{c}_i^L(p, u^i(p, x_i^a, t), u_x^i(p, x_i^a, t), v, t)
\end{aligned} \tag{2.98}$$

with $0 < t \leq t_s$, $i = 1, \dots, m_a - 1$. A transition condition of the i -th area either in Dirichlet or Neumann form depends on the time variable, the parameters to be estimated, and the solution of the neighboring area. Again the user may omit any of these functions, if a transition condition does not exist at a given x_i^a -value. More complex implicit boundary and transition condition can be defined in form of coupled algebraic equations.

Since the starting time is assumed to be zero, initial conditions must have the form

$$u^i(p, x, 0) = u_0^i(p, x) \ , \quad i = 1, \dots, m_a \tag{2.99}$$

and are defined for all $x \in [x_{i-1}^a, x_i^a]$, $i = 1, \dots, m_a$. If initial values for algebraic variables are not consistent, i.e., do not satisfy the algebraic equations of (2.96), the given values can be used as starting values for solving the corresponding system of nonlinear equations by Newton's method.

If the partial differential equations are to be coupled to ordinary differential algebraic equations, we proceed from an additional DAE-system of the form

$$\dot{v}_j = G_j(p, u^{i_j}(p, x_j, t), u_x^{i_j}(p, x_j, t), u_{xx}^{i_j}(p, x_j, t), v, t) \tag{2.100}$$

for $j = 1, \dots, n_{dc}$, and

$$0 = G_j(p, u^{i_j}(p, x_j, t), u_x^{i_j}(p, x_j, t), u_{xx}^{i_j}(p, x_j, t), v, t) \tag{2.101}$$

for $j = n_{dc} + 1, \dots, n_c$, confer (2.92). Initial values are

$$v(p, 0) = v_0(p) \tag{2.102}$$

that may depend again on the parameters to be estimated. The system has n_c components, i.e., $v = (v_1, \dots, v_{n_c})^T$. Coupling of ordinary differential equations is allowed at arbitrary points within the integration interval and the corresponding area is denoted by the index i_j . The spatial variable value x_j , some or all of them may coincide, belongs to the i_j -th area, i.e. $x_j \in [x_{i_j-1}^a, x_{i_j}^a)$ or $x_j \in [x_{m_a-1}^a, x_{m_a}^a]$, respectively, $j = 1, \dots, n_c$, and is called coupling point.

Coupling points are rounded to their nearest line when discretizing the system. The right-hand side of the coupling equation may depend on the corresponding solution of the partial equation and its first and second derivative subject to the space variable at the coupling point under consideration.

To indicate that the fitting criteria $h_k(p, t)$ depend also on the solution of the differential equation at the corresponding fitting point, where k denotes the index of a measurement set, we use the notation

$$h_k(p, t) = \bar{h}_k(p, u^{i_k}(p, x_k, t), u_x^{i_k}(p, x_k, t), u_{xx}^{i_k}(p, x_k, t), v(p, t), t) \quad (2.103)$$

and insert \bar{h}_k instead of h_k into the data fitting function. Again, the fitting criteria may depend on solution values at a given spatial variable value within an integration interval defined by the index i_k . The spatial variable x_k belongs to the i_k -th integration area, i.e. $x_k \in [x_{i_k-1}^a, x_{i_k}^a)$ or $x_k \in [x_{m_a-1}^a, x_{m_a}^a]$, respectively, $k = 1, \dots, r$, where r denotes the total number of measurement sets. The fitting criterion may depend on the solution of the partial equation and its first and second derivative with respect to the space variable at the fitting point. Fitting points are rounded to their nearest line when discretizing the system.

Basically, each integration area is treated as an individual boundary value problem, and is discretized separately by the method of lines. The transition functions are treated in the same way as Dirichlet or Neumann boundary conditions, respectively.

In order to achieve smooth fitting criteria and constraints, we assume that all model functions depend continuously differentiable on the parameter vector p . Moreover, we assume that the discretized system of differential equations is uniquely solvable for all p with $p_l \leq p \leq p_u$. A collection of 20 examples of partial differential equations that can be solved by the presented approach, and comparative numerical results are found in Schittkowski [433].

Example 2.23 (HEAT_B) *To illustrate different integration areas and the application of transition conditions, we again consider the heat equation, but now formulated over two areas,*

$$\begin{aligned} u_t^1 &= D_1 u_{xx}^1, & 0 < x \leq 0.5, \\ u_t^2 &= D_2 u_{xx}^2, & 0.5 < x \leq 1 \end{aligned} \quad (2.104)$$

with diffusion coefficients D_1 and D_2 , $t > 0$. The initial heat distribution at $t = 0$ is

$$\begin{aligned} u^1(p, x, 0) &= a \sin(\pi x) & 0 < x \leq 0.5, \\ u^2(p, x, 0) &= a \sin(\pi x) & 0.5 < x \leq 1, \end{aligned} \quad (2.105)$$

	p^*	p_0	p
D_1	1.0	5.0	1.077
D_2	1.0	0.2	0.974
a	1.0	10.0	1.029
b	2.0	1.0	1.966

Table 2.13: Exact, Start, and Computed Solution

and Dirichlet boundary values

$$u^1(p, 0, t) = u^2(p, 1, t) = 0 \quad (2.106)$$

for all $t \geq 0$ are supposed. First, we try to obtain a smooth transition from the first to the second area for $D_1 = D_2 = 1$ and $a = 1$, and define

$$u^1(p, 0.5, t) = b u^2(p, 0.5, t) \quad (2.107)$$

with $b = 1$ as the only transition condition between both areas. The surface plot is shown in Figure 2.30. We get a continuous transition, but not a smooth one. However, there is only one transition condition, the solution is not uniquely determined, and the resulting solution is more or less arbitrarily generated. If we require in addition that

$$u_x^2(p, 0.5, t) = u_x^1(p, 0.5, t) \quad , \quad (2.108)$$

we get the same solution as displayed in Figure 2.24.

Next, we construct a non-continuous transition between both areas by $b = 2$. We require that the flux is continuous, i.e., that the spatial derivatives coincide at the transition line. Then we construct a data fitting problem by computing simulated experimental data as done in the previous sections. Time values are $t_1 = 0.05, t_2 = 0.1, \dots, t_9 = 0.5$, spatial values are $x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$, and $x_4 = 0.8$. The 36 data simulated subject to the parameter values of Table 2.13 are perturbed by an error of 5 %.

Each integration area of the partial differential equation is discretized by 21 lines, and a fifth-order difference formula is applied. The resulting ODE system is solved by an implicit method with integration accuracy 10^{-6} . The least squares algorithm DFNLP of Schittkowski [429] with termination accuracy 10^{-10} stops after 47 iterations at the optimal solution, see Table 2.13 for results. Obviously, we are able to identify the parameters, also the parameter of the transition condition, within the accuracy of the experimental data. The final surface plot of the solution $u(p, x, t)$ is shown in Figure 2.31.

By the subsequent example, we want to outline the possibility to define transition conditions also for algebraic equations. Another reason for presenting the example is to show that our approach allows to integrate also time-independent systems of partial differential equations, i.e., systems that do not contain any time derivatives at all.

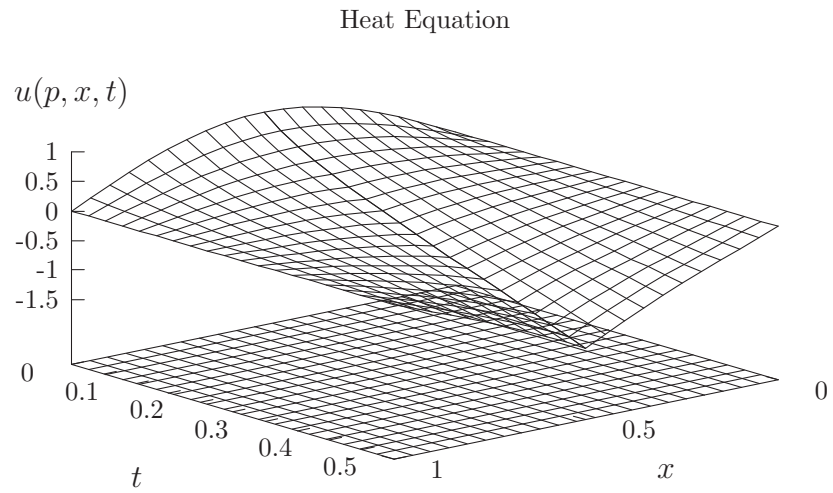


Figure 2.30: Continuous Transition Condition

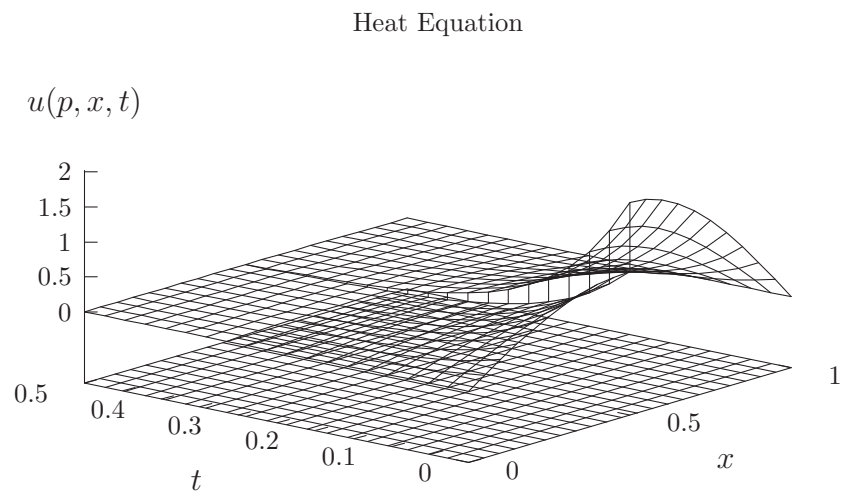


Figure 2.31: Non-Continuous Transition Condition

	p^*	p_0	p_l	p	p_u
k	1.0	0.5	0.959	1.004	1.049
a_1	1.0	0.5	0.920	1.002	1.084
a_2	1.0	0.5	0.943	1.004	1.064

Table 2.14: Exact Values, Starting Values, and Confidence Intervals

Example 2.24 (BEAM2) *Two beams are clamped at two end point and linked at an intermediate point by some kind of joint. Bending is modelled by a fourth-order differential equation*

$$E_I \frac{\partial^4}{\partial x^4} w + kw = f(x, t) \quad , \quad (2.109)$$

see Timoshenko and Goodier [507], where the load function f is given by

$$f(x, t) = -(a_1 t + a_2 t^2)(x^2 + (x - 3)^2) \quad .$$

$E_I = 0.01$ is the constant for flexural frigidty, and k , a_1 , and a_2 are parameters we want to identify. The spatial variable x varies between 0 and 3, where the transition is defined at $x_1^a = 1.2$. Initial conditions are $w(p, x, 0) = w_{xx}(p, x, 0) = 0$, and also the boundary conditions are set to zero, i.e.,

$$w(p, 0, t) = w(p, 3, t) = w_{xx}(p, 0, t) = w_{xx}(p, 3, t) = 0 \quad .$$

The joint condition requires that the solution values $w(p, x, t)$ and the third derivatives $w_{xxx}(p, x, t)$ coincide at $x = x_1^a$.

Experimental data are simulated at 10 time values 0.2, 0.4, ..., 2.0, and 9 spatial values 0.3, 0.4, ..., 2.7, perturbed subject to an error of 5 %. Exact, starting, and final parameter values are shown in Table 2.14 together with 5 % confidence intervals. The two integration areas are discretized with respect to 21 and 25 lines, and a five-point difference formula for first and second derivatives. The integration is performed with termination tolerance 10^{-5} . DFNLDP computes the solution after 10 iterations with final accuracy 10^{-11} . Figure 2.32 shows the final state variable w .

2.6.6 Switching Points

We consider now the same model that was developed so far in the previous sections, i.e., a system of one-dimensional partial differential algebraic equations with flux functions, coupled ordinary differential equations, and different integration areas with transition conditions, see (2.96). For the same reasons pointed out in Section 2.4.3, we suppose that n_b break or switching points with

$$\tau_0 = 0 < \tau_1 < \dots < \tau_{n_b} < \tau_{n_b+1} = T \quad (2.110)$$

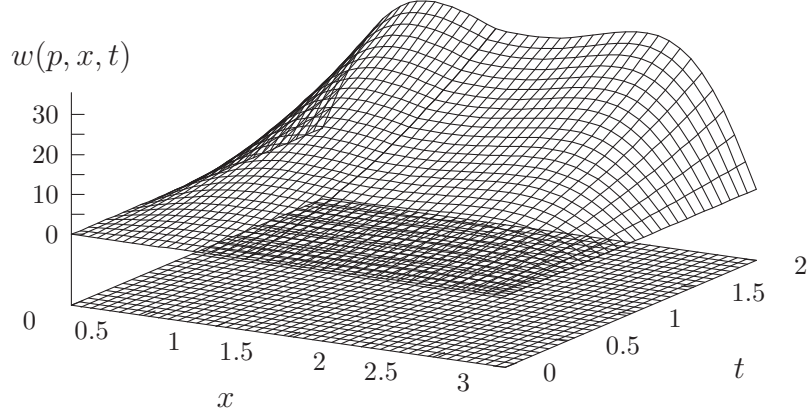


Figure 2.32: Third-Order Transition Condition

are given, where T is the last experimental time value.

For the first integration interval, the same initial, boundary, and transition values are given as before, see (2.99), (2.101), (2.97), and (2.98), respectively. But for all subsequent intervals the integration subject to the time variable is to be restarted at a switching point and new function values can be provided that may depend now also on the solution of the previous section. Initial values at a switching point are evaluated from

$$\begin{aligned} u^i(p, x, \tau_k) &= b_k^i(p, u_-^i(p, x, \tau_{k-1}), v_-(p, \tau_{k-1}), x) , \\ v(p, \tau_k) &= \tilde{b}(p, v_-(p, \tau_{k-1})) \end{aligned} \quad (2.111)$$

for $i = 1, \dots, m_a$, and $k = 1, \dots, n_b$, where $u_-^i(p, x, \tau_{k-1})$ and $v_-(p, \tau_{k-1})$ denote the solution of the coupled PDAE system in the previous time interval.

Since the right-hand side of the partial differential equation (2.96) and also the corresponding boundary and transition functions depend on the time variable, they may change from one interval to the next. Also non-continuous transitions at switching points are allowed.

It is possible that break points become variables to be adapted during the optimization process, as outlined in Example 2.8. However, there exists a very dangerous situation when a variable switching point passes or approximates a measurement time value during an optimization run. If both coincide and if there is a non-continuous transition, then the underlying model function is no longer differentiable with respect to the parameter to be optimized. Possible reactions of the least squares algorithm are slow final convergence rates

or break downs because of internal numerical difficulties. On the other hand, variable switching points are very valuable when trying to model the input feed of chemical or biological processes given by a bang-bang control function or any other one with variable break points.

Example 2.25 (HEAT B) *To show that our approach allows simultaneous fit of parameters defining non-continuous transitions with respect to time and space variables, we extend Example 2.23. First we get the time-dependent partial differential equation in two areas*

$$\begin{aligned} u_t^1 &= D_1 u_{xx}^1, & 0 < x \leq 0.5, \\ u_t^2 &= D_2 u_{xx}^2, & 0.5 < x \leq 1, \end{aligned} \quad (2.112)$$

see also (2.104), with diffusion coefficients D_1 and D_2 and $t > 0$. The initial heat distribution for $t = 0$ is the same as before, but a switching point τ_1 is introduced, where we assume that a certain value α is to be added to the system, i.e.

$$\begin{aligned} u^1(p, x, 0) &= a \sin(\pi x) & 0 < x \leq 0.5, \\ u^2(p, x, 0) &= a \sin(\pi x) & 0.5 < x \leq 1, \\ u^1(p, x, \tau_1) &= u_-^1(p, x, \tau_1) + \alpha & 0 < x \leq 0.5, \\ u^2(p, x, \tau_1) &= u_-^2(p, x, \tau_1) + \alpha & 0.5 < x \leq 1, \end{aligned} \quad (2.113)$$

see (2.105), where $u_-^1(p, x, t)$ and $u_-^2(p, x, t)$ denote the solution in the time interval $0 \leq t < \tau_1$. In a similar way, we adapt the Dirichlet boundary values

$$\begin{aligned} u^1(p, 0, t) &= 0, & 0 \leq t < \tau_1, \\ u^2(p, 0, t) &= 0, & 0 \leq t < \tau_1, \\ u^1(p, 0, t) &= \alpha, & \tau_1 \leq t, \\ u^2(p, 0, t) &= \alpha, & \tau_1 \leq t, \end{aligned} \quad (2.114)$$

see (2.106). Transition functions between both areas are the same as before, i.e.

$$\begin{aligned} u^1(p, 0.5, t) &= b u^2(p, 0.5, t), \\ u_x^2(p, 0.5, t) &= u_x^1(p, 0.5, t), \end{aligned} \quad (2.115)$$

see also (2.115) and (2.108). Experimental data are generated in same way as for Example 2.23 subject to parameter values of Table 2.13.

As before, each integration area of the partial differential equation is discretized by 21 lines, a fifth order difference formula is applied, the discretized system of 39 ODE's is solved by an implicit method with a restart at $t = \tau_1$, and the least squares algorithm DFNLP is started with the same solution tolerances. After 109 iterations the optimal solution listed in Table 2.15 is obtained, see Figure 2.33 for the final surface plot. Obviously, we are able to

	p^*	p_0	p
D_1	1.0	5.0	1.09287
D_2	1.0	0.2	0.97516
a	1.0	10.0	1.02542
b	2.0	1.0	1.97548
α	0.2	1.0	0.19998
τ_1	0.25	0.29	0.25184

Table 2.15: Exact, Start, and Computed Solution

Heat Equation

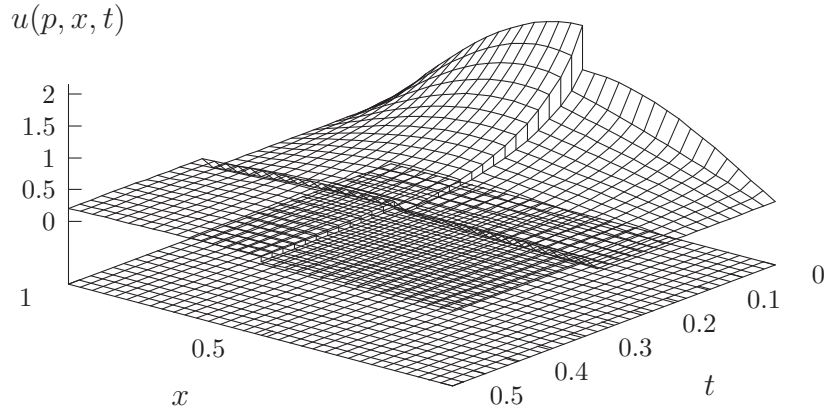


Figure 2.33: Non-Continuous Transition Condition and Switching Point

identify the system parameters, the parameter defining the jump in the transition condition, the switching time and the jump with respect to the time variable simultaneously within the accuracy provided by the experimental data.

However, we have to be careful when defining the switching time. Since the exact value $\tau_1 = 0.25$ is also an experimental time value, we have to avoid that the least squares objective function becomes non-differentiable at the optimal solution. Thus, the corresponding experimental data are omitted, and some smaller lower and upper bound are defined for the optimization variable, i.e. $0.21 \leq \tau_1 \leq 0.29$.

2.6.7 Constraints

In the previous sections, we extended the general partial differential equation (2.70) step by step and got fitting criteria $h_k(p, t)$ of the form

$$h_k(p, t) = \bar{h}_k(p, u^{i_k}(p, x_k, t), u_x^{i_k}(p, x_k, t), u_{xx}^{i_k}(p, x_k, t), v(p, t), t) \quad , \quad (2.116)$$

where $u^{i_k}(p, x_k, t)$ denotes the solution of the partial differential algebraic equation (2.82) in the i_k -th area, see also (2.102). $v(p, t)$ is the solution of a coupled system of ordinary differential algebraic equations, see (2.92), and x_k is the spatial variable value, where measurements are available, rounded to the closest line.

Thus, the data fitting problem consists of minimizing one of the the objective functions

$$\sum_{k=1}^r \sum_{i=1}^{l_t} (w_i^k (h_k(p, t_i) - y_i^k))^2 \quad (2.117)$$

for the least squares norm,

$$\sum_{k=1}^r \sum_{i=1}^{l_t} w_i^k |h_k(p, t_i) - y_i^k| \quad (2.118)$$

for the L_1 -norm, and

$$\max_{k=1, \dots, r; i=1, \dots, l_t} w_i^k |h_k(p, t_i) - y_i^k| \quad (2.119)$$

for the maximum-norm, where measurements (t_i, y_i^k) are given, $i = 1, \dots, l_t$, $k = 1, \dots, r$.

As for ordinary differential equations, additional nonlinear equality and inequality constraints of the form

$$\begin{aligned} g_j(p) &= 0, \quad j = 1, \dots, m_e \quad , \\ g_j(p) &\geq 0, \quad j = m_e + 1, \dots, m_r \end{aligned} \quad (2.120)$$

are allowed. These restrictions are useful to describe certain limitations on the choice of parameter values, for example monotonicity.

Example 2.26 (ISOTHERM2) *We consider the identification of a nonlinear coefficient function in a transport process through porous media, see Igler, Totsche, and Knabner [232] or Igler and Knabner [231]. The advective-dispersive transport equation is*

$$u_t = \frac{Du_x x - qu_x}{\phi(u) + 1} \quad (2.121)$$

for $0 \leq t \leq 6$ and $0 \leq x \leq 1$. For the diffusion coefficient, we choose $D = 0.1$ and the Darcy flow velocity is set to $q = 1$. $\phi(u)$ is the sorption isotherm we want to identify. Initial mass concentration is zero, $u(p, x, 0) = 0$, and Neumann boundary conditions are set,

$$u_x(p, 0, t) = \frac{q}{D}(u - f(t)) \quad ,$$

$$u_x(p, 1, t) = 0$$

with inflow concentration $f(t)$ given by linear interpolation of the data

i	p_i^0	p_i
1	0.5	0.565
2	0.7	0.565
3	0.8	0.751
4	0.9	0.879

Table 2.16: Starting and Computed Parameter Values

t	$f(t)$
0.0	1.0
1.0	0.5
2.0	0.1
3.0	0.0
10.0	0.0

Experimental measurement values are generated at 20 equidistant time values between 0 and 6, spatial value $x_1 = 1$, and a sorption isotherm $\phi(u) = \sqrt{u}$ we would like to identify. Subsequently uniformly distributed random errors (5 %) are added to the data, and we replace $\phi(u)$ in (2.121) by a piecewise linear interpolation of p_1, \dots, p_4 , see subsequent table.

u	$\phi(u)$
0.0	0
0.2	p_1
0.4	p_2
0.6	p_3
0.8	p_4

We know that u does never exceed 0.8 in this case. Constraints are defined to guarantee that the parameters remain monotone, that

$$p_1 \leq p_2 \leq p_3 \leq p_4 \leq 1 \quad .$$

The least squares code DFNLP is executed, where the underlying PDE is discretized at 21 lines. Starting from the data given in the subsequent table, DFNLP terminates after 6 iterations. The obtained optimal parameter values are listed in Table 2.16. Obviously the constraints are satisfied and the first one becomes active.

The deviation of $\phi(u)$ from the exact coefficient function \sqrt{u} is shown in Figure 2.34. We conclude that an identification of $\phi(u)$ is possible within the known experimental errors. Figure 2.35 shows that the experimental data are fitted and Figure 2.36 plots the surface of the solution function $u(p, x, t)$.

It is also possible to define dynamical constraints, where the restriction functions depend on the solution of the partial differential equation and its first and second spatial derivatives

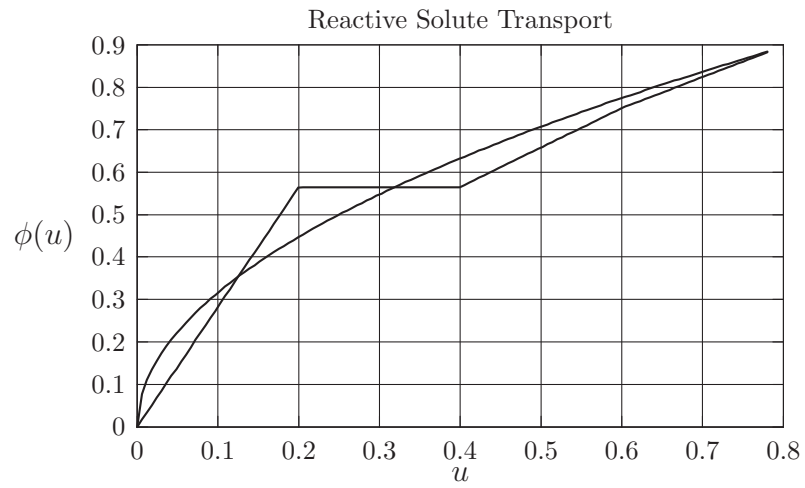


Figure 2.34: Identification Error

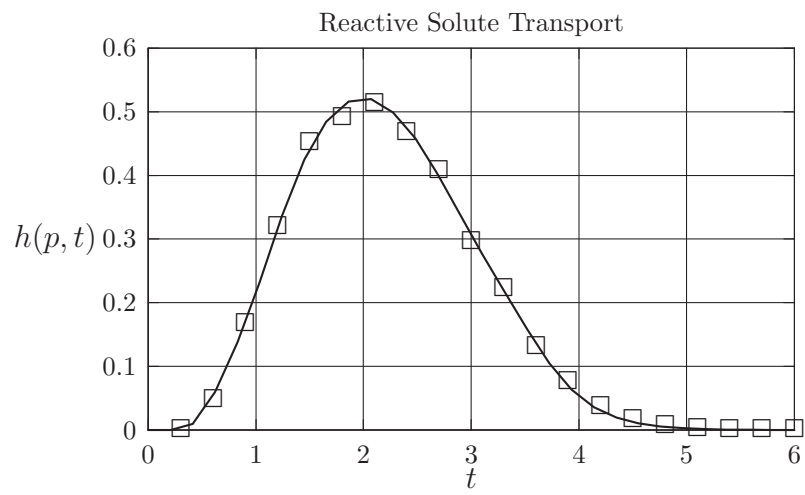


Figure 2.35: Function and Data Plot

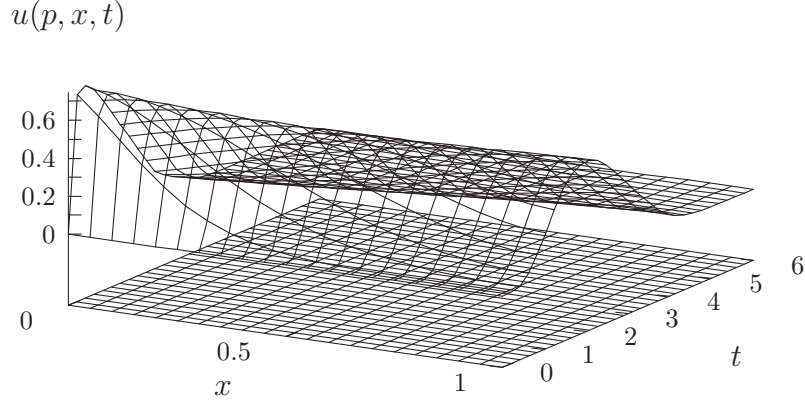


Figure 2.36: State Variable

at predetermined time and spatial values, and the solution of the coupled ordinary differential equation, that is,

$$g_j(p) = \bar{g}_j(p, u^{i_j}(p, x_j, t_{k_j}), u_x^{i_j}(p, x_j, t_{k_j}), u_{xx}^{i_j}(p, x_j, t_{k_j}), v(p, t_{k_j})) \quad (2.122)$$

for $j = m_e + 1, \dots, m_r$. Here the index i_j denotes the corresponding integration area that contains the spatial parameter x_j rounded to its nearest line, and k_j the corresponding experimental time where a restriction is to be formulated. Thus, constraints are evaluated only at certain lines and experimental time values. If they are required also at some intermediate points, one has to increase the number of lines or the number of experimental data with zero weights.

Equation (2.122) is the discretized form of the dynamical constraints we want to define. In a more general context, our intention is to limit certain functions depending on the state variable for all time and/or spatial variable value by

$$\bar{g}_j(p, u(p, x, t), u_x(p, x, t), u_{xx}(p, x, t), v(p, t), x, t) \geq 0 \quad (2.123)$$

or

$$\bar{g}_j(p, u^{i_j}(p, x_j, t), u_x^{i_j}(p, x_j, t), u_{xx}^{i_j}(p, x_j, t), v(p, t), t) \geq 0 \quad (2.124)$$

or

$$\bar{g}_j(p, u(p, x, t_j), u_x(p, x, t_j), u_{xx}(p, x, t_j), v(p, t_j), x) \geq 0, \quad (2.125)$$

respectively, for $j = m_e + 1, \dots, m_r$, $x \in [x_L, x_R]$, and $t \geq 0$. Of course these constraints may be mixed with the time-independent parameter constraints (2.120). Note that the

formulation of dynamical equality constraints does not make sense, since they should be treated as algebraic equations.

Example 2.27 (HEAT_R) *We consider again our standard test problem Example 2.17, the heat equation*

$$u_t = Du_{xx} \ ,$$

with diffusion coefficient D . The spatial variable x varies from 0 to 1, and the time variable is non-negative, i.e., $t \geq 0$. Initial heat distribution for $t = 0$ is

$$u(p, x, 0) = a \sin(\pi x)$$

for all $x \in (0, 1)$. Dirichlet boundary values are the same as before, confer (2.80), and the parameter vector $p = (D, a)^T$ is to be estimated subject to the same simulated experimental data computed for Example 2.17 with exact solution $p^ = (1, 1)^T$ and 21 discretization lines. Now we consider the second spatial derivatives $u_{xx}(p, x, t)$, see Figure 2.37. The biggest curvature is observed at the point $t = 0$ and $x = 0.5$ with $u_{xx}(p, x, t) \approx -10$. Our goal is to prevent the curvature from achieving any value below -8 . Thus, we formulate one dynamical constraint*

$$g(p) = u_{xx}(p, x, 0) + 8 \geq 0$$

or, after a suitable discretization,

$$g_j(p) = u_{xx}(p, x_j, 0) + 8 \geq 0$$

for $j = 1, \dots, 9$ with $x_j = 0.1j$. Note that the constraints are violated at the exact solution $p^ = (1, 1)^T$, for which experimental data are simulated.*

Starting from $p_0 = (2, 2)^T$, DFNLP computes the solution $p = (0.92, 0.81)^T$ in 10 iterations with termination accuracy 10^{-8} . The fifth constraint becomes active, i.e., $g_5(p) = 0.17 \cdot 10^{-13}$, and the remaining ones are satisfied, see Figure 2.38.

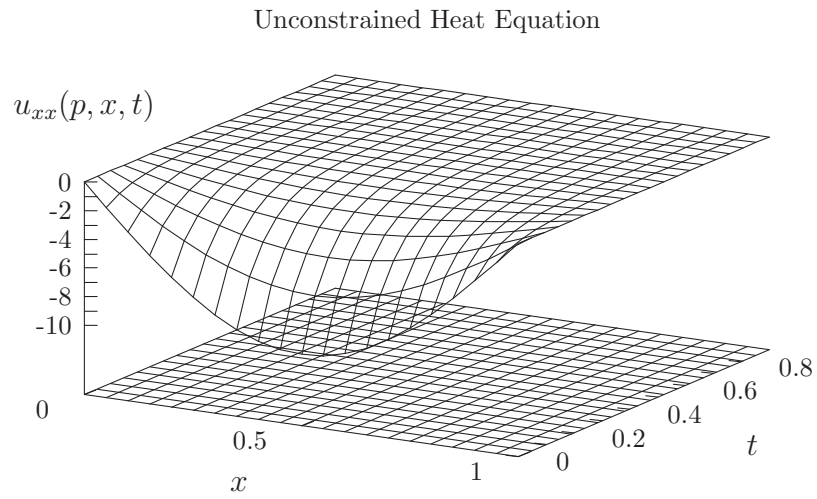


Figure 2.37: Second Partial Derivatives at p^*

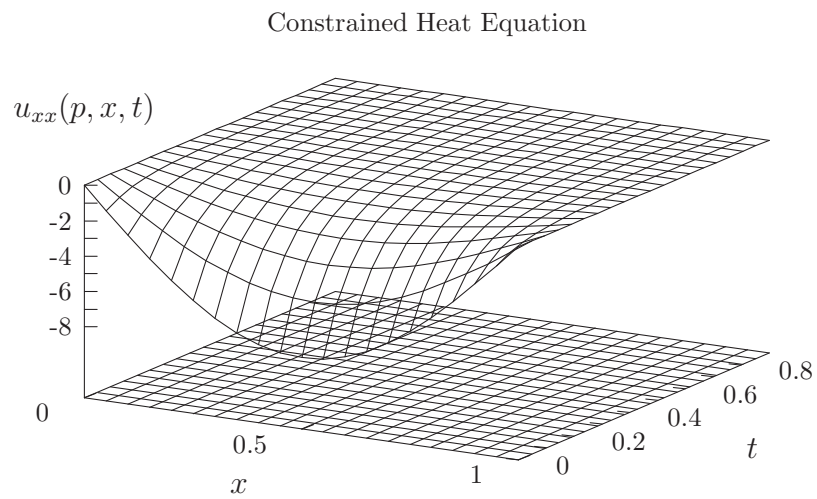


Figure 2.38: Second Partial Derivatives at Optimal Solution p

2.7 Optimal Control Problems

The main difference between optimal control and data fitting problems is the formulation of the objective function. In the first case, the objective function to be minimized is an arbitrary function depending on the solution of the underlying dynamical system, often formulated as an integral, whereas in the second case, we minimize a sum of squares or any related norm, as considered in the previous sections.

Another difference is that optimal control models contain so-called control variables in addition to some discrete parameters, that is a function $s(t) \in \mathbb{R}^{n_s}$ to be modified until a suitable cost function is minimized. However, we have to approximate control variables so that they can be represented by a final set of parameters. Typical control approximations are piecewise constant, piecewise linear, cubic spline, or exponential spline functions either subject to a constant or a variable set of break points. In the latter case, the switching points are also optimization variables, where the required serial order leads to a couple of additional linear inequality constraints. A special form of control variable is called *bang-bang* function, i.e., functions defined by two constant alternative values, where only the switching points are variable.

The underlying dynamical system to be considered, is now exactly the same considered so far for our data fitting applications. It is assumed that either an ordinary differential, a differential algebraic equation, a one-dimensional time-dependent partial differential, or a partial differential algebraic equation is given.

It is out of the scope of this software documentation to present a broad introduction into theory and numerical algorithms for optimal control or applications. For more detailed information, especially also about numerical algorithms and available software in case of ordinary differential equations, see e.g. Goh and Teo [174], Jennings, Fisher, Teo, Goh [236], Bulirsch and Kraft [64], or Machielsen [312]. For solving optimal control problems based on distributed parameter systems, see Ahmed and Teo [4], Neittaanmäki and Tiba [351], Blatt and Schittkowski [49], or Birk et al. [44].

Our intention is, to consider only control problems that can be solved by the data fitting approach studied so far. Thus, we suppose for example that we want to minimize a certain quadratic or L_2 norm

$$J_1(p, s) = \int_{x_L}^{x_R} (f_1(p, s(\bar{t}), u(p, x, \bar{t}), u_x(p, x, \bar{t}), u_{xx}(p, x, \bar{t}), v(p, \bar{t}), x) - \bar{f}_1(x))^2 dx \quad (2.126)$$

by fixing a time value $0 \leq \bar{t} \leq T$, and where integration is performed over all areas, confer (2.96). $\bar{f}_1(x)$ is given and our goal is to minimize the distance of a certain criterion f_1 from a_0 by adapting some parameters p and the control function $s(t)$. It is assumed that the control variable depends only on the time variable as in most practical applications. Otherwise, one could try to exchange the role of t and x .

Vice versa, it is possible that the cost function is evaluated at a fixed spatial value \bar{x} , so

that the integration is performed over the total time horizon

$$J_2(p, s) = \int_0^T (f_2(p, s(t), u^k(p, \bar{x}, t), u_x^k(p, \bar{x}, t), u_{xx}^k(p, \bar{x}, t), v(p, t), t) - \bar{f}_2(t))^2 dt \quad (2.127)$$

with a given time-dependent function $\bar{f}_2(t)$. f_2 may depend on solution values at a given spatial variable value \bar{x} in an integration interval defined by the index k . Thus, \bar{x} belongs to the k -th integration area, $\bar{x} \in [x_{k-1}^a, x_k^a)$ or $\bar{x} \in [x_{m_a-1}^a, x_{m_a}^a]$, respectively. As a special case (2.127) contains control problems depending only on the solution $v(p, t)$ of a system of ordinary differential algebraic equation of the form (2.36) with initial values.

Another possibility is that there is only one function to be minimized with fixed time and spatial values, say

$$J_3(p, s) = f_3(p, s(T), u^k(p, \bar{x}, T), u_x^k(p, \bar{x}, T), u_{xx}^k(p, \bar{x}, T), v(p, T)) \quad (2.128)$$

Without loss of generality the fixed time value is supposed to be the final integration time T . As a special case we get a time-minimal optimal control problem

$$J_3(p, s) = T \quad ,$$

where we minimize only the final integration time.

In the first two situations we may discretize the integral at certain spatial or time values, and get immediately a least squares formulation

$$J_1(p, s) \approx \sum_{j=1}^{n_x} w_j (f_1(p, s(\bar{t}), u^{k_j}(p, x_j, \bar{t}), u_x^{k_j}(p, x_j, \bar{t}), u_{xx}^{k_j}(p, x_j, \bar{t}), v(p, \bar{t}), x_j) - \bar{f}_1(x_j))^2 \quad (2.129)$$

with suitable weights w_j . The index k_j denotes the corresponding spatial integration interval containing x_j .

In the second case, the integral is replaced by

$$J_2(p, s) \approx \sum_{i=1}^{n_t} w_i (f_2(p, s(t_i), u^k(p, \bar{x}, t_i), u_x^k(p, \bar{x}, t_i), u_{xx}^k(p, \bar{x}, t_i), v(p, t_i), t_i) - \bar{f}_2(t_i))^2 \quad , \quad (2.130)$$

where k denotes again the spatial integration area containing \bar{x} .

Even in the third situation, we are able to get a trivial least squares formulation

$$J_3(p, s) \approx \left(\sqrt{f_3(p, s(T), u^k(p, \bar{x}, T), u_x^k(p, \bar{x}, T), u_{xx}^k(p, \bar{x}, T), v(p, T))} \right)^2 \quad , \quad (2.131)$$

if we knew that the algorithm we apply to solve the least squares problem, is capable to handle situations where the length of the sum of squares is one.

Nonlinear state and control constraints may be added to the optimal control problem either in explicit form (2.50) for the discrete parameter vector p , or in form of dynamical

inequality constraints (2.52) and (2.123), respectively, where we have to add the dependency from the control variable, say

$$\bar{g}_j(p, s(t), u^{i_j}(p, x_j, t), u_x^{i_j}(p, x_j, t), u_{xx}^{i_j}(p, x_j, t), v(p, t), t) \geq 0 \quad . \quad (2.132)$$

The discretized formulation leads to

$$g_j(p) = \bar{g}_j(p, s(t_j), u^{i_j}(p, x_j, t_j), u_x^{i_j}(p, x_j, t_j), u_{xx}^{i_j}(p, x_j, t_j), v(p, t_j), t_j) \quad (2.133)$$

for $j = m_e + 1, \dots, m_r$. Here the index i_j denotes the corresponding integration area that contains the spatial parameter x_j rounded to its nearest line, and t_j a suitable discrete time value where a restriction is to be formulated.

Many practical control problems are given in form of higher order differential equations with boundary conditions. However, arbitrary nonlinear equality constraints with respect to the optimization parameters can be added to the optimal control problem. Thus, an equivalent optimal control problem with boundary values is easily formulated, see Section 2.4.6 and especially Example 2.15.

Example 2.28 (B_BLOCK) *By the first example we consider the optimal control of an ordinary differential equation, where we know the exact control function. The goal is to find an optimal distribution of a drug, more precisely a β -blocker, so that a given concentration level is followed as closely as possible, see Cherruault [90]. The underlying pharmaceutical system is modeled by two kinetic differential equations*

$$\begin{aligned} \dot{x}_1 &= -(k_{12} + k_e)x_1 + k_{21}x_2 + s(t), & x_1(0) &= a \quad , \\ \dot{x}_2 &= k_{12}x_1 - k_{21}x_2, & x_2(0) &= 0 \quad . \end{aligned} \quad (2.134)$$

Initial substrate concentration is $a = 18$. The transfer coefficients could have been obtained by a previous data fitting run, see Section 2.4.1, and are given by $k_{12} = 2.9545$, $k_{21} = 5.7214$, and $k_e = 0.3658$, see Cherruault [90]. Control function $s(t)$ is piecewise constant with switching points $1/n$ for $n = 10$, $n = 20$, and $n = 40$, respectively, and function values s_1, \dots, s_n . The distance of $x_1(t)$ from the given goal function $\bar{f}_2(t) = a$ is evaluated at 40 equidistant points between 0 and 1. The numerical solution of the ODE is restarted at each switching point.

DFNLP is started with termination tolerance 10^{-15} from zero control values, where an implicit solver is applied to integrate the ODE with final accuracy 10^{-6} . Table 2.17 contains numbers of iterations n_{it} and final residual values r for $n = 10, 20, 40$. Figures 2.39 to 2.41 show the substrate distribution $x_1(t)$ over the time axis, Figure 2.42 the corresponding one for $x_2(t)$, and Figure 2.43 the exact control function $s^(t) = ak_e + ak_{12}\exp(-k_{21}t)$ and the approximated ones.*

If we apply non-continuous control functions as in the previous example, it is necessary to restart the integration at each switching point, to avoid the generation of extremely

n	n_{it}	r
10	63	$0.43 \cdot 10^{-5}$
20	92	$0.34 \cdot 10^{-6}$
30	108	$0.47 \cdot 10^{-10}$

Table 2.17: Performance Results

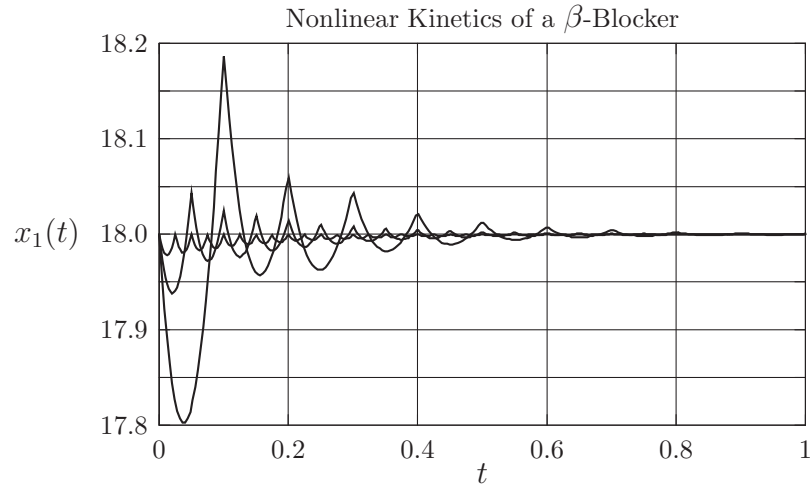


Figure 2.39: State Variable $x_1(t)$ for $n = 10$

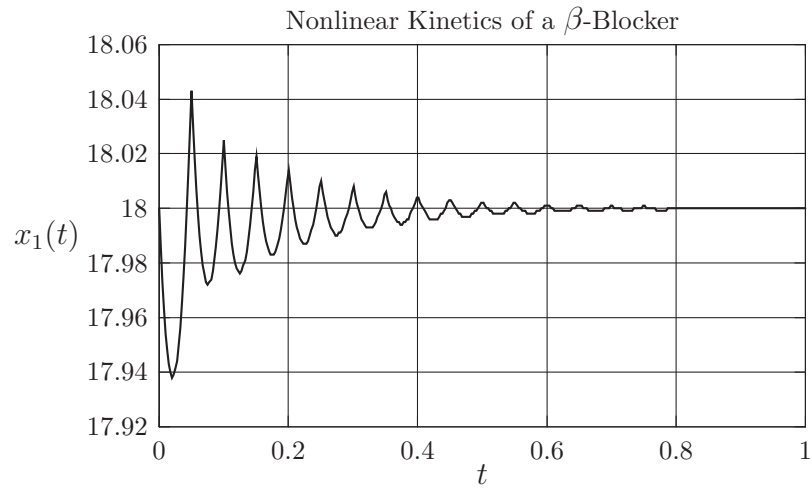


Figure 2.40: State Variable $x_1(t)$ for $n = 20$

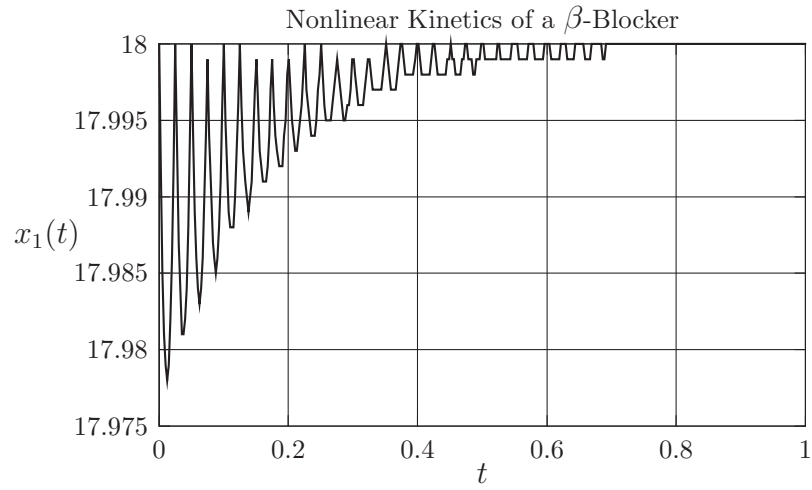


Figure 2.41: State Variable $x_1(t)$ for $n = 40$

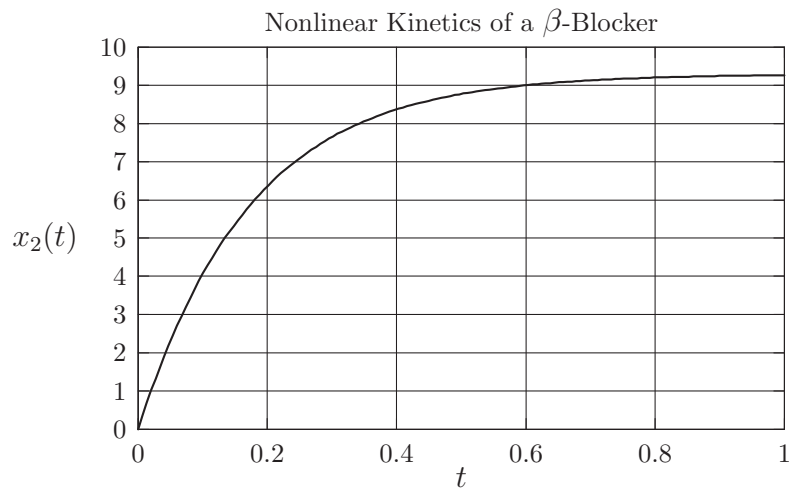


Figure 2.42: State Variable $x_2(t)$

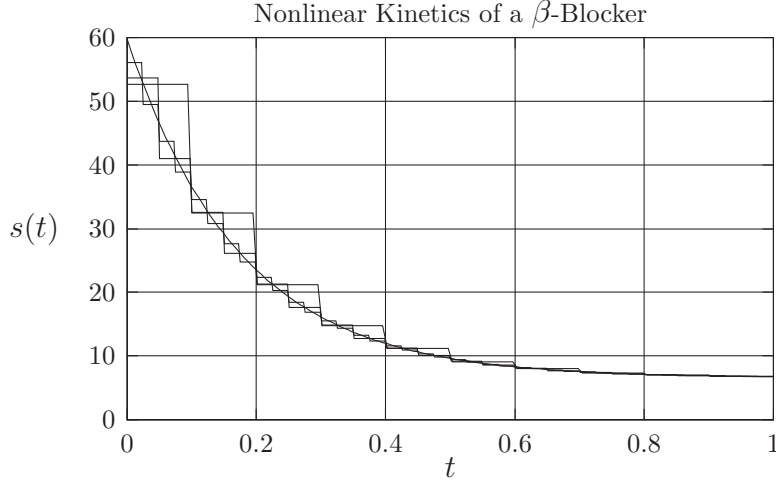


Figure 2.43: Computed and Exact Control Variables for $n = 10, 20, 40$

short steps of the ODE solver. As mentioned above, these switching points may become optimization parameters. A special situation arises, if we consider bang-bang controls, where control function values jump from one constant level to another one. By the next example we illustrate the usage of a bang-bang control, the approximation of a boundary function, and the possibility to minimize also the final integration time in case of a partial differential state equation.

Example 2.29 (TIME-OPT) *We consider now another variant of our standard test problem Example 2.17, the heat diffusion model. On the one hand, we want to approximate a given final boundary function $\bar{f}_1(x)$ at $t = T$ as closely as possible, on the other the final time T is to become as small as possible. Thus, the problem is a mixture of a minimum-norm and a time-optimal one, see Schittkowski [424]. A constant scaling parameter α is introduced to weight the two different objectives, and we get the objective function*

$$J(T, s) = \int_0^1 (u(T, s, x, T) - \bar{f}_1(x))^2 dx + \alpha T \quad (2.135)$$

subject to the state equation

$$\begin{aligned} u_t(T, s, x, t) &= u_{xx}(T, s, x, t) , \\ u_x(T, s, 0, t) &= 0 , \\ u(T, s, x, 0) &= 0 , \\ u(T, s, 1, t) + u_x(T, s, 1, t) &= s . \end{aligned} \quad (2.136)$$

For our numerical test we choose $\alpha = 0.01$ and $\bar{f}_1(x) = 0.5 - 0.5x^2$. The control variable s is a bang-bang functions jumping from 1 to -1 and vice versa at some switching points s_1, \dots, s_5 .

i	p_i^0	p_i
1	0.2	0.2914
2	0.4	0.2914
3	0.6	0.7268
4	0.8	0.7268
5	0.9	0.8483

Table 2.18: Initial and Final Switching Points

To be able to apply our data fitting software under consideration, we normalize the time variable to get a state equation defined for $0 \leq t \leq 1$. The differential equation in (2.136) becomes

$$u_t = Tu_{xx} \quad . \quad (2.137)$$

Then the least squares objective function to be minimized, is

$$h(T, s, t) = \sum_{i=1}^9 (u(T, s, x_i, 1) - \bar{f}_1(x_i))^2 + (\sqrt{\alpha T})^2 \quad (2.138)$$

Moreover, we have to take into account additional linear constraints

$$0 \leq p_1 \leq \dots \leq p_5 \leq 1$$

for the switching points.

When starting NLPQLP from the starting values of Table 2.18 with termination tolerance 10^{-8} , 21 discretization lines, and an implicit ODE solver with final accuracy 10^{-6} , we get the results of Table 2.18 after 8 iterations. Obviously some of the switching points coincide in agreement with the results of Schittkowski [424]. Final integration time is $T = 1.329$. The maximum deviation of $u(p, x, T)$ from $\bar{f}_1(x)$ at a grid point is 0.0113. State and control variables are displayed in Figure 2.44 and 2.45, respectively.

Heat Diffusion

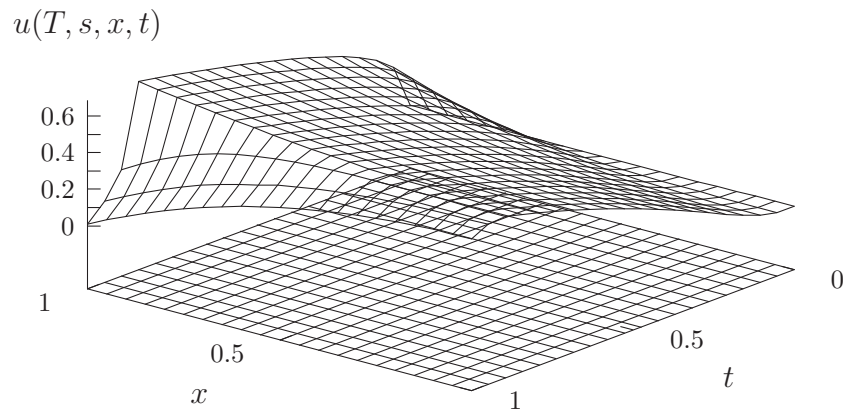


Figure 2.44: State Variable

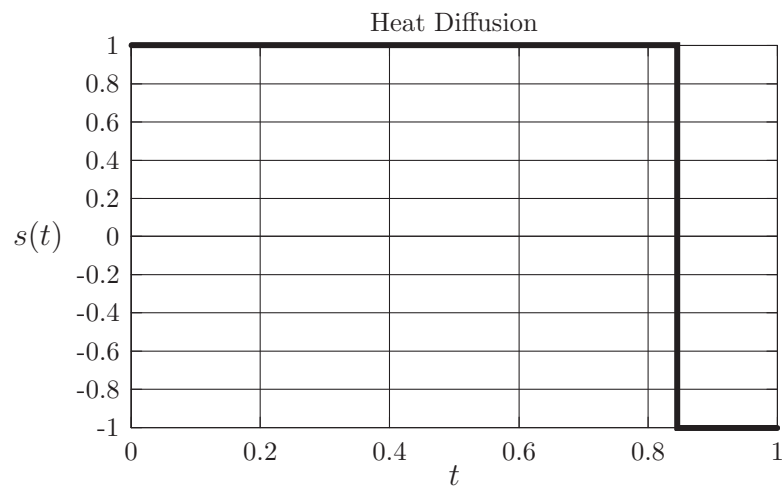


Figure 2.45: Control Variable

Chapter 3

Statistical Analysis and Experimental Design

It is outside the scope of this documentation to present a review of statistical methods that are available today to analyze data and results. There exists a broad area in statistics, called nonlinear regression or parameter estimation, where these techniques are developed in detail, see for example the books of Bard [27], Beck and Arnold [32], Draper and Smith [124], Gallant [162], Ratkowsky [395], Seber [457], Seber and Wild [458], or Ross [411].

The first question is how to get suitable confidence intervals for the estimated parameters. This is one of the main investigations when analyzing the output of data fitting. Related questions are whether it is possible at all to identify parameters, or how to eliminate redundant ones, as will be discussed in the subsequent sections. For these and related modeling and simulation techniques, see also the books of Walter and Pronzato [538] and of van den Bosch and van der Klauw [515].

Another important question is experimental design, where we want to create or improve existing experimental conditions. The goals are to reduce the number of costly experiments, to reduce error variances, or to get identifiable parameters. Typically, initial values of differential equations or control functions e.g. for input feeds are to be adapted, see e.g. Winer, Brown and Michels [551] or Ryan [414].

The standard tool to analyze the statistical properties of a dynamical model is the evaluation of confidence intervals based on some simplifying assumptions. The confidence region subject to a given significance level is an ellipsoid which is typically approximated by a surrounding box. Whereas small interval lengths can be interpreted as well-identifiable parameters, larger intervals could be due to degenerated ellipsoids. A more rigorous analysis is given in Section 3.1.

In many practical situations, dynamical models contain too many parameters which are difficult to estimate simultaneously, i.e., are overdetermined. An important question is how to detect the relative significance of parameters and how to eliminate redundant ones based on a given experimental design. A heuristic approach is presented in Section 3.2, which is

computationally attractive and easy to implement. The idea is to analyze eigenvalues and eigenvectors of the covariance matrix. The absolutely largest coefficient of the eigenvector belonging to the biggest eigenvalue is eliminated and marks a less significant parameter. The procedure is repeated until some criteria are satisfied, e.g., reaching a certain significance tolerance. The result is a serial order of parameters according to their relevance, and which helps to decide which parameters could be eliminated or whether additional experiments should be performed. Section 3.2 contains some illustrative examples and a more realistic data fitting problem based on a chemical reaction of an isothermal reactor with too many model parameters.

So far we proceeded from a given experimental design and try to fit some model parameters. However, the initial design might not be the best one and the question is how to improve or even optimize it. Possible design parameters are time dependent input feeds, initial concentrations or temperatures. The goal is to construct a suitable performance criterion depending on design parameters, additional constraints as far as necessary, and to solve the resulting nonlinear optimization problem. Since the confidence intervals mentioned above are mainly determined by the diagonal elements of the covariance matrix, a possible objective function is the trace of this matrix, see Section 3.3. Special emphasis is given to the efficient computation of derivatives, where first and second order derivatives of the model function of our dynamical system are all approximated by forward differences. To show that this approach is nevertheless a quite stable and efficient procedure if carefully implemented, two examples are included. The first one is a microbial growth model, see Banga et al. [22], which consists of a small system of only three differential equations, two model parameters and design parameters in form of initial concentrations and an input feed. However, one of the model parameters is extremely difficult to estimate and the authors decided to apply a stochastic search method. The other example is intensively investigated by Bauer et al. [29], the reaction of urethane. The model consists of three differential and three algebraic equations, and becomes more complex because of additional nonlinear equality and inequality constraints.

There remains the question how the techniques described in Section 3.3, can also be used for locating experimental time values. Especially in case of time expensive experiments, it is highly desirable to minimize their number and to conduct experiments only within relevant time intervals. Thus, we apply the same strategy outlined before, but add artificial weight factors to the observations at a predefined, relatively dense grid specified by the user. These weights are considered then as design parameters. A particular advantage is that derivatives subject to weights are obtained without additional computational efforts. Section 3.4 contains the corresponding analysis and again the urethan example, now with the aim to reduce the number of experiments.

The techniques described so far do not depend on any special structure of the mathematical model. The only assumption is smoothness of objective function and constraints, i.e., these functions should be twice continuously differentiable subject to the model and design parameters. All examples with practical background consist of ordinary differential

or differential algebraic equations, since the imbedded solution process generates additional numerical noise making numerical results and conclusions more realistic. The techniques and part of the examples are also discussed in Schittkowski [441]. A case study for a system of partial differential equations is found in Schittkowski [447].

3.1 Confidence Intervals

We proceed from a general nonlinear model in its simplest form

$$\eta = h(p, t) + \epsilon \quad , \quad (3.1)$$

see also (1.1), where we omit another possible dependency of the right-hand side from the solution of a dynamical system without loss of generality. $h(p, t)$ is our model function depending on a set of model parameters $p \in \mathbb{R}^n$ and $t \in \mathbb{R}$ is the independent model variable, also called explanatory or regression variable. The function $h(p, t)$ is supposed to be differentiable subject to $p \in \mathbb{R}^n$, and at least continuous with respect to t . It is assumed that there is a true parameter value p^* , which is unknown and which is to be estimated by a least squares fit. The response $\eta \in \mathbb{R}$ is the dependent model variable.

The above formulation proceeds for simplicity from a scalar variable t . Generalizations to multi-dimensional regression variables are possible without loss of generality. Also multi-response models where η possesses arbitrary dimension, can be considered, see Seber and Wild [458].

To estimate the true, but unknown parameter value p^* from given experimental data t_i and y_i , $i = 1, \dots, l$, we minimize the least squares function

$$s(p) = \sum_{i=1}^l (h(p, t_i) - y_i)^2 \quad (3.2)$$

over all $p \in \mathbb{R}^n$. Let \hat{p} denote the solution of this data fitting problem. Then \hat{p} is also called the ordinary least squares estimator (OLS) to distinguish it from alternative techniques, for example from the weighted or generalized least squares estimators. The question we are interested in is how far away \hat{p} is from the true parameter p^* .

It is assumed that the independent model values t_i are given a priori without errors, and that ϵ_i denotes the statistical error of the measurements or the response variable, respectively. As usual, we suppose that the errors $\epsilon_i = y_i - h(p^*, t_i)$ are independent and normally distributed with mean value zero and known constant variance σ^2 , i.e., $\epsilon_i \sim N(0, \sigma^2)$ for $i = 1, \dots, l$.

The basic idea is to linearize the nonlinear model in a neighborhood of p^* and to apply linear regression analysis, since linear models are very well understood, see Seber [456]. By defining

$$f(p) = (h(p, t_1), \dots, h(p, t_l))^T$$

and $\epsilon = y - f(p^*)$, $q = p - p^*$, $\epsilon = (\epsilon_1, \dots, \epsilon_l)^T$, $y = (y_1, \dots, y_l)^T$, we get from the first-order Taylor expansion

$$\begin{aligned} s(p) &= \|f(p) - y\|^2 \\ &\approx \|f(p^*) + \nabla f(p^*)^T(p - p^*) - y\|^2 \\ &= \|\nabla f(p^*)^T q - \epsilon\|^2 . \end{aligned}$$

Here $\|\cdot\|$ denotes the Euclidian norm. We denote by $F^* = \nabla f(p^*)$ the Jacobian matrix of $f(p)$ at $p = p^*$, and assume that F^* has full rank. A solution of the linear least squares problem is immediately obtained from the normal equations

$$\hat{q} = (F^* F^{*T})^{-1} F^* \epsilon ,$$

from which we get a first-order approximation of the solution \hat{p} by

$$\hat{p} = p^* + (F^* F^{*T})^{-1} F^* \epsilon .$$

From this approximation, some statistical properties known for linear models can be derived also for nonlinear ones. Under additional regularity assumptions, see Seber and Wild [458], \hat{p} and $s^2 = s(\hat{p})/(l - n)$ are consistent estimates of p^* and σ^2 , that means they converge with probability 1 to the true values, and are asymptotically normally distributed as l goes to infinity. Moreover, we know that due to the normal distribution of the errors, \hat{p} is also a maximum likelihood estimator.

The error in parameters, $\hat{p} - p^*$, is approximately normally distributed with mean value 0 and covariance matrix $\sigma^2 I^{*-1}$, where I^* is defined by $I^* = F^* F^{*T}$. In addition, the expression

$$\frac{1}{ns^2} (\hat{p} - p^*)^T I^* (\hat{p} - p^*)$$

follows the F -distribution with $(n, l - n)$ degrees of freedom within the linearization error. Thus, an approximate $100(1 - \alpha)\%$ confidence region for p^* is given by the set

$$\{\bar{p} : (\bar{p} - \hat{p})^T \hat{I} (\bar{p} - \hat{p}) \leq ns^2 F_{n, l-n}^\alpha\} , \quad (3.3)$$

where $\hat{I} = \nabla f(\hat{p}) \nabla f(\hat{p})^T$ estimates I^* . This result is very similar to the corresponding confidence region for linear models.

For a numerical implementation, however, (3.3) is inconvenient. To get individual confidence intervals for the coefficients of p^* , we consider an arbitrary linear combination $a^T p$. It is possible to show that approximately

$$\frac{a^T \hat{p} - a^T p^*}{s \sqrt{a^T I^{*-1} a}} \sim t_{l-n} , \quad (3.4)$$

where t_{l-n} is the t -distribution with $l - n$ degrees of freedom. A $100(1 - \alpha)\%$ confidence interval is then given by

$$\left[a^T \hat{p} - t_{l-n}^{\alpha/2} s \sqrt{a^T I^{*-1} a} , a^T \hat{p} + t_{l-n}^{\alpha/2} s \sqrt{a^T I^{*-1} a} \right] . \quad (3.5)$$

When setting $a = e_i$ for $i = 1, \dots, n$ successively, where e_i is the i -th unit vector, and when estimating I^* by \hat{I} , we get the approximate confidence intervals

$$\left[\hat{p}_i - t_{l-n}^{\alpha/2} s \sqrt{\hat{d}_{ii}}, \hat{p}_i + t_{l-n}^{\alpha/2} s \sqrt{\hat{d}_{ii}} \right] \quad (3.6)$$

for the i -th individual model parameter value p_i^* , $i = 1, \dots, n$. In this case, \hat{p}_i is the i -th coefficient of \hat{p} and \hat{d}_{ii} the i -th diagonal element of \hat{I}^{-1} , see also Gallant [161] or Donaldson and Schnabel [119].

However, (3.6) is valid only approximately depending on the quality of the linearization or the curvature of $f(p)$, respectively. Donaldson and Schnabel [119] present some examples, where the confidence intervals are very poor. Thus, we have to be very careful when computing (3.6) without additional linearization checks.

Example 3.1 (PARID15/30/60/120) *We consider the model function*

$$h(p, t) = \frac{p_1 p_3}{p_1 - p_2} (e^{-p_2 t} - e^{-p_1 t})$$

with three unknown parameters $p = (p_1, p_2, p_3)^T$ to be estimated. First, we define a true parameter value $p^* = (0.1, 1, 100)^T$ and generate experimental data sets in the following way. For $l = 120, l = 60, l = 30$, and $l = 15$, we evaluate $y_i = h(p^*, t_i) + \epsilon_i$, where ϵ_i is a normally distributed error with variance $\sigma^2 = 0.01$, at equidistant grid points t_i within the interval $[0, 60]$, $i = 1, \dots, l$. Then we solve the corresponding data fitting problem (3.2) starting from $p_0 = (0.05, 2, 120)^T$ with termination accuracy 10^{-10} .

Subsequently, we compute the confidence intervals (3.6) as outlined above for the significance level $\alpha = 1\%$, see Table 3.1. The corresponding lower and upper bounds and the computed parameter vectors are \hat{p}_i^l , \hat{p}_i , and \hat{p}_i^u , $i = 1, 2, 3$. Moreover, we determine $s^2 = s(\hat{p})/(l - n)$, an estimate of the variance σ^2 . Parameter p_1^* can be estimated for all sample sizes quite successfully. The variance estimates converge to the true value, as expected. Figure 3.1 shows the fitted data for $l = 120$ measurements.

3.2 Significance Levels by Eigenvalue/-vector Analysis of the Fisher Information Matrix

Proceeding from a parameter estimation model, corresponding data, and a successful least squares fit, significance levels of the estimated parameters are to be evaluated. If a model seems to be overdetermined, i.e., contains too many parameters compared to the number of equations, the levels give an impression of the significance of parameters and help to decide upon questions like

Table 3.1: Confidence Intervals for Different Sample Sizes

	$l = 120$	$l = 60$	$l = 30$	$l = 15$
\hat{p}_1^l	0.0998	0.0963	0.0945	0.0891
\hat{p}_1	0.0998	0.0996	0.0989	0.0973
\hat{p}_1^u	0.1025	0.1032	0.1036	0.1055
\hat{p}_2^l	0.942	0.918	0.889	0.281
\hat{p}_2	0.986	0.997	1.042	0.977
\hat{p}_2^u	1.030	1.076	1.196	1.673
\hat{p}_3^l	93.7	91.4	87.6	22.7
\hat{p}_3	98.5	100.1	104.2	98.6
\hat{p}_3^u	103.3	108.8	120.8	174.4
s^2	0.0105	0.0106	0.0087	0.0081

- which parameters can be identified,
- which parameters can be treated as constants,
- whether additional experimental should be added or not.

Moreover, overdetermined data fitting problems lead to unstable and slow convergence of Gauss-Newton-type least squares algorithms with a large number of iterations until termination tolerances are satisfied.

We have seen in the previous section that $\hat{s}^2 \hat{I}^{-1}$ can be considered as an approximation of the covariance matrix $\sigma^2 I^{\star-1}$, where

$$\hat{s}^2 = s(p) = \frac{1}{l-n} \sum_{i=1}^l (h(\hat{p}, t_i) - y_i)^2 ,$$

see (3.2), $\hat{I} = \nabla f(\hat{p}) \nabla f(\hat{p})^T$, and \hat{p} a least squares estimate for the true, but unknown parameter p^* . Assumptions are independent and normally distributed errors in the measurements with mean value 0 and variance σ^2 .

A more rigorous analysis based on the maximum-likelihood function leads to the theorem of Cramér and Rao, which states that the inverse of the Fisher information matrix is a lower bound for the covariance matrix of the parameter errors. This matrix is approximately given by

$$\hat{I}_F = \frac{1}{\hat{s}^2} \nabla f(\hat{p}) \nabla f(\hat{p})^T . \quad (3.7)$$

For a precise definition of this matrix and a proof see e.g. Goodwin and Payne [178].

Since all induced matrix norms are greater than the spectral radius of a matrix, we apply the L_2 -norm, i.e.,

$$\|\hat{I}_F^{-1}\|_2^2 = |\lambda_{\max}(\hat{I}_F^{-1})| = \frac{1}{|\lambda_{\min}(\hat{I}_F)|} , \quad (3.8)$$

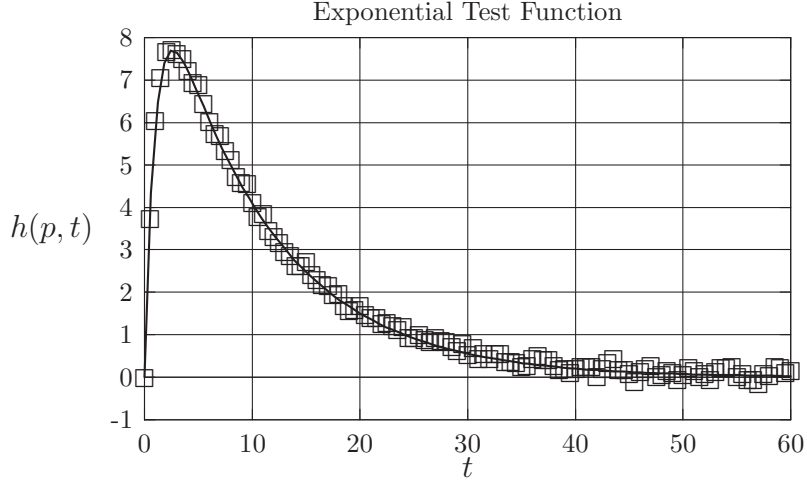


Figure 3.1: Model Function and Data

λ_{max} and λ_{min} denote the largest and smallest eigenvalue of a matrix, respectively. Since small eigenvalues of \hat{I}_F enforce large entries of the covariance matrix, we try to reduce them by successive elimination of parameters corresponding to large eigenvector coefficients. The order by which the variables are eliminated, can be considered as an indication about their relative significance, the highest level reflects the highest priority.

We proceed from a given significance tolerance $\gamma > 0$, known experimental data, and an optimal solution \hat{p} of the corresponding least squares data fitting problem. We try to satisfy

$$\|\hat{I}_F^{-1}\|_2 = \sqrt{\frac{1}{|\lambda_{min}(\hat{I}_F)|}} < \gamma . \quad (3.9)$$

Assuming a sufficiently accurate approximation of p^* , the true parameter vector, we hope to get sufficiently small variances.

Note that very small or zero eigenvalues lead to the conclusion that some parameters cannot be estimated at all by the underlying model and the available data, or that there are combinations of highly correlated parameters, see Caracotsis and Stewart [75]. To detect the significant parameters on the one hand and the redundant or dependent parameters on the other we apply the subsequent procedure, see also Schneider, Posten, and Munack [451] or Majer [315]. The idea is to successively eliminate parameters until (3.9) is satisfied. The cycle is terminated in one of the following situations:

1. The smallest eigenvalue of the Fisher information matrix is smaller than a threshold value, see (3.9).
2. The parameter correlations are significantly reduced, e.g., by 25 %.
3. None of the above termination reasons are met and all parameters have been eliminated.

Algorithm 3.1 Let $k = 1$, $\hat{J}_0 = \emptyset$, $\hat{I}_F^k = \frac{1}{\hat{s}^2} \nabla f(\hat{p}) \nabla f(\hat{p})^T$, \hat{x} minimizer of (3.2), and $\gamma > 0$ be given.

1. Compute the lowest eigenvalue λ_{\min} of \hat{I}_F^k and a corresponding eigenvector $v_{\min} \in \mathbb{R}^n$, $v_{\min} = (v_1^{\min}, \dots, v_n^{\min})^T$.
2. If $\lambda_{\min} > \frac{1}{\gamma^2}$, then stop. The required significance level is reached.
3. Determine j_0 with
$$|v_{j_0}| = \max_{1 \leq j \leq n} |v_j| ,$$
eliminate the j_0 -th row and column from \hat{I}_F^k , denote the resulting matrix by \hat{I}_F^{k+1} , and let $\hat{J}_{k+1} = \hat{J}_k \cup \{j_0\}$.
4. If $k = n - 1$ then stop, a further reduction is not possible.
5. Replace $k + 1$ by k and repeat from Step 1.

After termination, the indices in \hat{J}_k represent the significance levels of the parameters. Level 1 corresponds to the first eliminated variables, level 2 to the second, etc. The final level can be assigned to several parameters indicating a group of identifiable parameters. Possible conclusions are to add more experimental data or to fix some parameters for subsequent evaluations. Thus, the determination significance levels are part of the experimental design process to validate a parameter estimation model.

Example 3.2 (LKIN/_A3/_A4) A linear ordinary differential equation describes a kinetic process in the form

$$\begin{aligned} \dot{y}_1 &= -k_{11}y_1 & , \quad y_1(0) = D & , \\ \dot{y}_2 &= k_{11}y_1 - k_{22}y_2 & , \quad y_2(0) = 0 & . \end{aligned} \tag{3.10}$$

A 95 % confidence region as outlined in the previous section, is shown in Table 3.2, i.e., $\hat{c}_i = 2t_{l-n}^{\alpha/2} s \sqrt{\hat{d}_{ii}}$, see (3.6). The estimated error variance is $0.41 \cdot 10^{-3}$, the maximum correlation is 0.57, and the covariance values are sufficiently small, see also Figure 3.2.

Now we introduce some additional parameters with very severe internal dependencies,

$$\begin{aligned} \dot{y}_1 &= -\sqrt{k_{11}}k_{12}y_1 & , \quad y_1(0) = D_1 + 0.1D_2 & , \\ \dot{y}_2 &= k_{11}k_{12}y_1 - (k_{21} + 2k_{22})y_2 & , \quad y_2(0) = 0 & . \end{aligned} \tag{3.11}$$

The same statistical analysis as above leads to the significance intervals of Table 3.4. The correlation coefficients between k_{11} and k_{12} , between k_{21} and k_{22} , and between D_1 and D_2 are exactly 1. By successive elimination of parameters with highest coefficient of the eigenvector belonging to the lowest eigenvalue, see Table 3.3, priority levels are computed as shown in

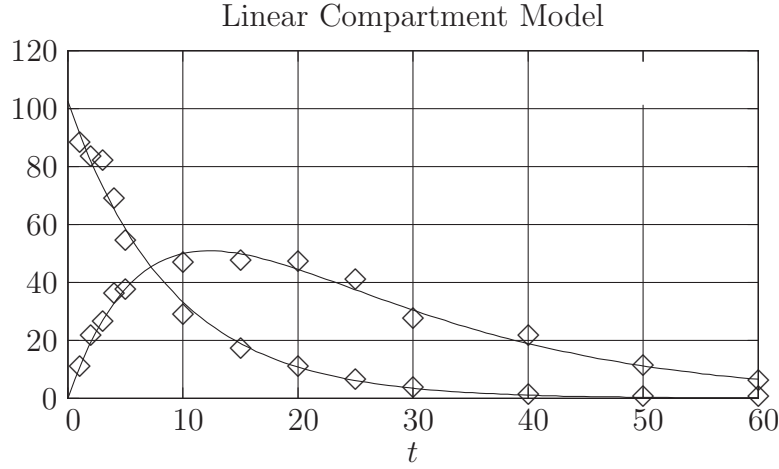


Figure 3.2: Model Functions and Data

Table 3.4. They exactly reflect the artificially generated dependencies. The parameters k_{11} , k_{22} , and D_1 obtained the highest scores and are considered as the most significant ones. We even observe that the influence of k_{22} on the solution is greater than that of k_{21} as can be expected from the different coefficients in (3.11). An important side effect is that the maximum correlation is reduced from 1.0 to 0.56.

Besides of detecting dependencies among parameters, the proposed analysis helps to find redundant ones, as shown by a slight modification of (3.10). An additional redundant parameter r is added to the first differential equation leading to a very small perturbation of the solution by choosing $\epsilon = 10^{-14}$,

$$\begin{aligned} \dot{y}_1 &= -k_1 y_1 + \epsilon r, & y_1(0) &= D, \\ \dot{y}_2 &= k_1 y_1 - k_2 y_2, & y_2(0) &= 0. \end{aligned} \quad (3.12)$$

The priority analysis detects the redundant parameter r , see Table 3.5. The starting value of the redundant parameter is not changed by the least squares algorithm.

Table 3.2: Confidence Intervals for (3.10)

p	\hat{p}_i	\hat{c}_i
k_1	0.1126	0.0034
k_2	0.0571	0.0022
D	102.4778	1.79

Table 3.3: Elimination of Parameters for (3.11)

k	$\lambda_{\min}(\hat{I}_F^k)$	$ v_{j_0} $	j_0
1	$-0.37 \cdot 10^{-11}$	0.89	3
2	$-0.12 \cdot 10^{-11}$	0.99	6
3	0.46.7	0.97	1
4	2578.3	1.0	5

Table 3.4: Confidence Intervals and Priority Levels for Overdetermined System (3.11)

p	\hat{p}_i	\hat{c}_i	\hat{J}_k
k_{11}	0.1328	0.0041	4
k_{12}	0.8476	0.00065	2
k_{21}	0.0314	0.00047	1
k_{22}	0.0128	0.00094	4
D_1	51.2396	0.96	4
D_2	51.2396	0.96	3

Example 3.3 (BATCH_F1/F2/F3/F4) *A practical example is the kinetic model of a chemical reaction system in an isothermal batch reactor, see Biegler, Damiano and Blau [41] or Majer [315],*

$$\begin{aligned}
\dot{x}_1 &= -k_2 x_2 x_8 \ , \\
\dot{x}_2 &= -k_1 x_2 x_6 + k_{-1} x_{10} - k_2 x_2 x_8 \ , \\
\dot{x}_3 &= k_2 x_2 x_8 + k_1 x_4 x_6 - 0.5 k_{-1} x_9 \ , \\
\dot{x}_4 &= -k_1 x_4 x_6 + 0.5 k_{-1} x_9 \ , \\
\dot{x}_5 &= k_1 x_2 x_6 + k_{-1} x_{10} \ , \\
\dot{x}_6 &= -k_1 x_2 x_6 - k_1 x_4 x_6 + k_{-1} x_{10} + 0.5 k_{-1} x_9 \ , \\
0 &= -x_7 + x_6 + x_8 + x_9 + x_{10} - Q^+ \ , \\
0 &= -x_8 (K_2 + x_7) + K_2 x_1 \ , \\
0 &= -x_9 (K_3 + x_7) + K_3 x_3 \ , \\
0 &= -x_{10} (K_1 + x_7) + K_1 x_5 \ .
\end{aligned} \tag{3.13}$$

We have

Table 3.5: Priority Levels for Redundant System (3.12)

p	\hat{p}_i	\hat{J}_k
k_1	0.1126	2
k_2	0.0571	2
D	102.4778	2
r	1.000	1

$$\begin{aligned}
Q^+ &= 0.0131 \ , \\
T_b &= 342.15 \ , \\
k_1 &= \exp \left(p_1 - \left(\frac{1}{T} - \frac{1}{T_b} \right) \exp(p_7) \right) \ , \\
k_2 &= \exp \left(p_2 - \left(\frac{1}{T} - \frac{1}{T_b} \right) \exp(p_8) \right) \ , \\
k_{-1} &= \exp \left(p_3 - \left(\frac{1}{T} - \frac{1}{T_b} \right) \exp(p_9) \right) \ , \\
K_1 &= \exp(-p_4) \ , \\
K_2 &= \exp(-p_5) \ , \\
K_3 &= \exp(-p_6) \ ,
\end{aligned}$$

nine parameters to be estimated, and three experimental data sets obtained under different temperatures and initial concentrations,

T	$x_1(0)$	$x_2(0)$	$x_3(0)$	$x_4(0)$	$x_5(0)$	$x_6(0)$
313.15	1.7066	8.3200	0.0100	0.0000	0.0	0.0131
340.15	1.6497	8.2200	0.0104	0.0017	0.0	0.0131
373.00	1.5608	8.3546	0.0082	0.0086	0.0	0.0131

In case of estimating only one data set, it is obvious that there are strong internal dependencies between p_1 and p_7 , p_2 and p_8 , and p_3 and p_9 . This is reflected by the priority listed in Table 3.6, where $\gamma = 0.1$. At least one of the two corresponding priorities obtained the lowest possible value. To improve the number of parameters which can be identified, we add up to two additional data sets, see again Table 3.6. For three different data sets, seven of nine parameters are considered as identifiable within one group similar to the results obtained by Majer [315]. We also observe that the parameter values get more and more stabilized, and some of them for three data sets are quite far away from the parameter values for one data set.

3.3 Experimental Design

Mathematical models describe the dynamical behavior of a system with the goal to allow numerical estimation of model parameters a user is interested in. These parameters identify

	$T = 340.15$		$T = 313.15,$ $T = 340.15$		$T = 313.15$ $T = 340.15,$ $T = 373$	
p	\hat{p}	\hat{J}_k	\hat{p}	\hat{J}_k	\hat{p}	\hat{J}_k
p_1	$2.8 \cdot 10^{-5}$	3	0.0	3	0.0	3
p_2	1.6	6	1.1	4	1.1	3
p_3	8.7	5	2.8	4	5.0	3
p_4	34.2	6	21.6	4	24.9	3
p_5	26.5	4	17.2	2	18.0	2
p_6	32.5	6	20.1	4	23.2	3
p_7	10.6	6	9.3	4	9.3	3
p_8	8.6	1	8.9	4	8.9	3
p_9	0.0002	2	0.04	1	0.04	1

Table 3.6: Parameter Values and Priorities for Example 3.3

the system under consideration, and are to be verified by experiments. However, the experimental design often depends on parameters which must be set in advance to be able to measure certain output data of an experiment. Examples are initial concentration of substrates, input feeds of a chemical reactor, temperature distributions, etc. In addition, our model may depend on universal physical parameters like gas constant, absolute temperature, or gravitational constant.

To determine the experimental design parameters in an optimal way, we first have to find a suitable guess for the model parameters either from the literature or some preliminary experiments. We have seen in the previous sections that the covariance matrix determines the confidence region of the model parameters, see (3.3). Since we have now additional freedom to design an experiment, we can use the design parameters to minimize the volume of the corresponding ellipsoid based on a suitable criterion.

To formalize the situation, we denote again the model parameters by $p \in \mathbb{R}^{n_p}$ and the design parameters by $q \in \mathbb{R}^{n_q}$. In case of a dynamic, i.e., time dependent parameters, for example a control function, we assume that the control function is approximated by finitely many parameters.

Now we extend our model function $h(p, t)$ by the design parameters, $h(p, q, t)$, and assume that we have a set of experimental time values t_k , $k = 1, \dots, l$. Moreover, we let

$$f(p, q) = (h(p, q, t_1), \dots, h(p, q, t_l))^T$$

and denote by $F(p, q) = \nabla_p f(p, q)$ the Jacobian matrix of $f(p, q)$ subject to $p \in \mathbb{R}^{n_p}$, where $q \in \mathbb{R}^{n_q}$. For simplicity, we assume that $F(p, q)$ has full rank for all p and q .

A formal performance measure is available based on the covariance matrix $C(p, q) = I(p, q)^{-1}$, where $I(p, q) = F(p, q)F(p, q)^T$ denotes an approximation of the Fisher information

matrix, and where we omit a guess for the error variances of the measurements to simplify the notation. In other words, we assume that all experimental data are measured with constant error. The volume of a confidence region for a given model parameter $p \in \mathbb{R}^{n_p}$ is given by

$$\{\bar{p} : (\bar{p} - p)^T I(p, q) (\bar{p} - p) \leq \alpha_{n_p}\} , \quad (3.14)$$

with a statistical parameter α_{n_p} , see (3.3).

Formula (3.14) describes an ellipsoid, and the goal is to minimize its volume on the one hand, but on the other to prevent also degenerate situations where the maximum and minimum eigenvalue drift away. This is to be achieved by adapting the design parameter q for a given model parameter p , which is obtained either from a preliminary experiment, literature, or a reasonable guess. Possible criteria are available either for $C(p, q)$ or $I(p, q)$, respectively, depending on the procedure how to measure or estimate the volume and the structure of the ellipsoid. The most popular ones are

$$\begin{aligned} D & : & \det(C(p, q)) \\ A & : & \text{trace}(C(p, q)) \\ A^* & : & -\text{trace}(I(p, q)) \\ E & : & \lambda_{\min}(I(p, q)) \\ E^* \text{ or } C & : & \lambda_{\min}(I(p, q)) / \lambda_{\max}(I(p, q)) \end{aligned}$$

Here $\lambda_{\min}(I(p, q))$ and $\lambda_{\max}(I(p, q))$ denote the minimum and maximum eigenvalues of $I(p, q)$. For a more detailed discussion, see, e.g., Winer, Brown and Michels [551] or Ryan [414].

For our numerical implementation, we use the A -criterion, since the computationally attractive confidence intervals by which the size of the ellipsoid is estimated, take only the diagonal elements of the covariance matrix into account, see (3.6). This leads for each $p \in \mathbb{R}^{n_p}$ to the optimization problem

$$\begin{aligned} & \min \text{trace}(C(p, q)) \\ q \in \mathbb{R}^{n_q} : & \quad g_j(p, q) = 0 \quad , \quad j = 1, \dots, m_e \quad , \\ & \quad g_j(p, q) = 0 \quad , \quad j = m_e + 1, \dots, m \quad , \\ & \quad q_l \leq q \leq q_u \quad , \end{aligned} \quad (3.15)$$

where we add additional bounds for the variables q and additional equality and inequality constraints depending on the given model parameters p and the design variables q to be computed.

There remains the question how to compute the derivatives of the objective function

$$\phi(q) := \text{trace}(C(p, q))$$

subject to q in an efficient way. Numerical differentiation of $\phi(q)$ subject to q by a difference formula based on a previous numerical differentiation of $h(p, q, t)$ subject to p by another difference formula is unstable because of accumulation of truncation errors. It is assumed that second order analytical partial or mixed-partial derivatives are not available. Thus, we try to find a reasonable compromise which nevertheless leads to sufficiently stable procedure.

Differentiation of the objective function of (3.15) subject to q_r , $1 \leq r \leq n_q$, gives

$$\begin{aligned}
\frac{\partial}{\partial q_r} \phi(q) &= \frac{\partial}{\partial q_r} (\text{trace}(C(p, q))) \\
&= \text{trace} \left(\frac{\partial}{\partial q_r} I(p, q)^{-1} \right) \\
&= \text{trace} \left(-I(p, q)^{-1} \frac{\partial}{\partial q_r} I(p, q) I(p, q)^{-1} \right) \\
&= -\text{trace} \left(I(p, q)^{-1} \frac{\partial}{\partial q_r} (F(p, q) F(p, q)^T) I(p, q)^{-1} \right) \\
&= -\text{trace} \left(I(p, q)^{-1} \left(\frac{\partial}{\partial q_r} F(p, q) F(p, q)^T \right. \right. \\
&\quad \left. \left. + F(p, q) \frac{\partial}{\partial q_r} F(p, q)^T \right) I(p, q)^{-1} \right) .
\end{aligned} \tag{3.16}$$

There remains differentiation of the $l \times n_p$ matrix

$$\begin{aligned}
\frac{\partial}{\partial q_r} F(p, q) &= \frac{\partial}{\partial q_r} \nabla_p f(p, q) \\
&= \left(\frac{\partial^2}{\partial q_r \partial p_i} h(p, q, t_k) \right)_{i=1, n_p; k=1, l} .
\end{aligned} \tag{3.17}$$

The mixed partial derivatives of the model function $h(p, q, t)$ subject to p and q are approximated by forward differences

$$\begin{aligned}
\frac{\partial^2}{\partial q_r \partial p_i} h(p, q, t_k) &\approx \frac{1}{\epsilon_r \epsilon_i} ((h(p + \epsilon_i e_i, q + \epsilon_r e_r, t_k) + h(p, q, t_k)) \\
&\quad - (h(p, q + \epsilon_r e_r, t_k) + h(p + \epsilon_i e_i, q, t_k)))
\end{aligned} \tag{3.18}$$

for $k = 1, \dots, l$ and $i = 1, \dots, n_p$. Here, $e_i \in \mathbb{R}^{n_p}$ and $e_r \in \mathbb{R}^{n_q}$ are the i -th and r -th unit vectors, respectively, and ϵ_i, ϵ_r are suitable perturbation tolerances, e.g., chosen by $\epsilon_i := \max(1, |p_i|)\epsilon$ and $\epsilon_r := \max(1, |q_r|)\epsilon$ with a certain tolerance $\epsilon > 0$ which must be selected very carefully.

Equation (3.18) is written in a form to show that cancelation appears only once. Since the evaluation of the objective function $\phi(q)$, i.e., of $F(p, q) = \nabla_p f(p, q)$, requires also an approximation of first derivatives of the form

$$\frac{\partial}{\partial p_i} h(p, q, t_k) \approx \frac{1}{\epsilon_i} (h(p + \epsilon_i e_i, q, t_k) - h(p, q, t_k)) \quad , \quad (3.19)$$

only two additional evaluation of h , i.e., $h(p + \epsilon_i e_i, q + \epsilon_r e_r, t_k)$ and $h(p, q + \epsilon_r e_r, t_k)$ are required to get the mixed second order derivatives (3.18).

The perturbation tolerance ϵ should not be chosen too small. Depending on the condition number of the information matrix, even large values like $\epsilon = 0.01$ or even $\epsilon = 0.1$ are applicable and lead to stable solution processes subject to a surprisingly small optimality criterion.

Example 3.4 (MICGROWX/Y) Banga et al. [22] consider a design problem based on an unstructured microbial growth model to determine feed rate profiles in fed-batch bio-reactors. They mention that numerical instabilities prevent application of gradient-based optimization procedures. Instead, they use a stochastic search algorithm.

The process is described by two differential equations and the integration of an input function $F_{in}(t)$,

$$\begin{aligned} \dot{C}_S &= -\sigma C_X + F_{in}(t) \frac{C_{Sin} - C_S}{V} \quad , \quad C_S(0) = C_S^0 \quad , \\ \dot{C}_X &= \mu C_X - F_{in}(t) \frac{C_X}{V} \quad , \quad C_X(0) = \frac{C_X^0}{V_0} \quad , \\ \dot{V} &= F_{in}(t) \quad , \quad V(0) = V_0 \quad , \end{aligned} \quad (3.20)$$

where

$$\begin{aligned} \mu &= \mu_m \frac{C_S}{K_p + C_S + C_S^2/K_i} \quad , \\ \sigma &= \frac{\mu}{Y_{XS} + m} \quad , \\ V_0 &= V^* \frac{C_{Sin}}{C_{Sin} - C_S^0} \end{aligned}$$

and $m = 0.29$, $C_{Sin} = 500$, $Y_{XS} = 0.47$, $V^* = 7$, and $\mu_m = 2.1$. F_{in} is an input control function chosen very close to the optimal solution found by Versyck [526], K_p , K_i are model parameters to be estimated, and initial values C_S^0 , C_X^0 are design parameters. It turns out that K_p is very difficult to estimate. Starting from some reasonable initial guesses $K_p = 10$, $K_i = 0.1$, $C_S^0 = 40$, $C_X^0 = 10$, artificial measurements are generated and perturbed by a uniform error of 5 %. Then, confidence intervals are computed for the design parameters K_p and K_i .

In the next step, we consider the feed controls at 19 grid points as additional design parameters, and 20 constraints are added to prevent that $C_S(t)$ falls below zero. The perturbation

tolerance for gradient approximations by forward differences is set to $\epsilon = 0.01$. NLPQLP needs 18 iterations to reduce the performance criterion from $1.3 \cdot 10^5$ to 0.009 under termination accuracy 10^{-8} . Optimal design parameters are the initial concentrations $C_S^0 = 38.9$ and $C_X^0 = 14.3$, and the optimal feed curve is shown in Figure 3.3. In Figures 3.4 and 3.5 the corresponding state functions $C_S(t)$ and $C_X(t)$ are plotted.

After getting the optimal design parameters, the confidence intervals are computed in the same way as for the starting values. Standard deviations are reduced from 239.8 to 0.093 for K_p and from 0.0096 to 0.0022 for K_i . Moreover, the correlation coefficient is reduced from 0.99 to 0.19.

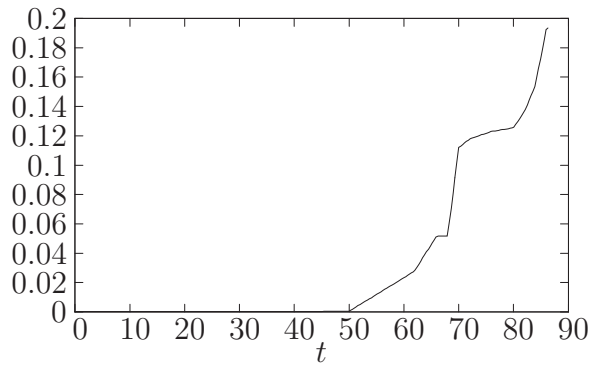


Figure 3.3: Control Function $F_{in}(t)$

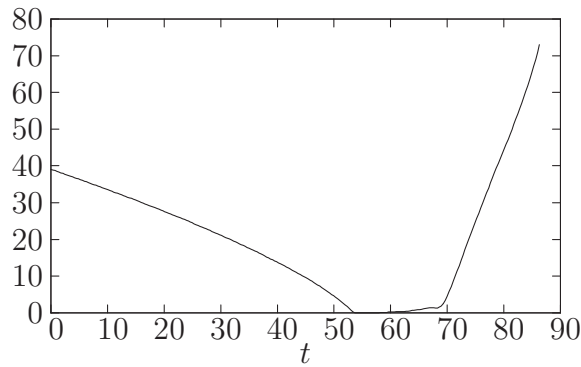


Figure 3.4: State Function $C_S(t)$

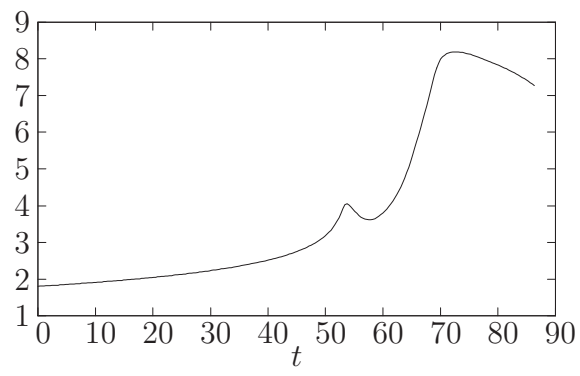


Figure 3.5: State Function $C_X(t)$

Example 3.5 (URETHAN1/2) A practically relevant example is studied by Bauer et al. [29], the reaction of urethane. The corresponding DAE describing the reaction of phenylisocyanate (n_1), butanol (n_2), urethane (n_3), allophante (n_4), and isocyanurate (n_5) consists of three differential and three algebraic equations of index 1,

$$\begin{aligned}
\dot{n}_3 &= V(r_1 - r_2 + r_3) , & n_3(0) &= 0 \\
\dot{n}_4 &= V(r_2 - r_3) , & n_4(0) &= 0 \\
\dot{n}_5 &= Vr_4 , & n_5(0) &= 0 \\
0 &= n_1 + n_3 + 2n_4 + 3n_5 - n_{a1} - n_{1ea}(t) , \\
0 &= n_2 + n_3 + n_4 - n_{a2} - n_{2eb}(t) , \\
0 &= n_6 - n_{a6} - n_{6ea}(t) - n_{6eb}(t) ,
\end{aligned} \tag{3.21}$$

where n_6 denotes the solvent and

$$\begin{aligned}
V &= \sum_{i=1}^6 \frac{M_i n_i}{\rho_i} , & k_1 &= k_{ref1} \exp(-E_{a1}(1/T(t) - 1/T_{ref1})/R) , \\
r_1 &= k_1 \frac{n_1 n_2}{V^2} , & k_2 &= k_{ref2} \exp(-E_{a2}(1/T(t) - 1/T_{ref2})/R) , \\
r_2 &= k_2 \frac{n_1 n_3}{V^2} , & k_3 &= k_2/k_c , \\
r_3 &= k_3 \frac{n_4}{V} , & k_4 &= k_{ref4} \exp(-E_{a4}(1/T(t) - 1/T_{ref4})/R) , \\
r_4 &= k_4 \frac{n_1^2}{V^2} , & k_c &= k_{c2} \exp(-d_{h2}(1/T(t) - 1/T_{g2})/R) .
\end{aligned}$$

Two input feeds are given in form of non-decreasing functions $feed_a(t)$ and $feed_b(t)$, $t \in [0, 80]$, and define $n_{1ea}(t) = n_{a1ea} feed_a(t)$, $n_{2eb}(t) = n_{a2eb} feed_b(t)$, $n_{6ea}(t) = n_{a6ea} feed_a(t)$, and $n_{6eb}(t) = n_{a6eb} feed_b(t)$. Mol ratios, active ingredients, and the initial volume have to satisfy certain bound constraints,

$$\begin{aligned}
0.1 &\leq MV_1 \leq 10 , \\
0 &\leq MV_2 \leq 1000 , \\
0 &\leq MV_3 \leq 10 , \\
0 &\leq g_a \leq 0.8 , \\
0 &\leq g_{aea} \leq 0.9 , \\
0 &\leq g_{aeb} \leq 1 , \\
0 &\leq V_a \leq 0.00075 ,
\end{aligned}$$

and are connected to the remaining parameters by analytical equations

$$\begin{aligned}
MV_1(n_{a1} + n_{a1ea}) &= n_{a2} + n_{a2eb} , \\
MV_2 n_{a1} &= n_{a1ea} , \\
MV_3 n_{a1} &= n_{a2eb} , \\
g_a(n_{a1}M_1 + n_{a2}M_2 + n_{a6}M_6) &= n_{a1}M_1 + n_{a2}M_2; , \\
g_{aea}(n_{a1ea}M_1 + n_{a6ea}M_6) &= n_{a1ea}M_1 , \\
g_{aeb}(n_{a2eb}M_2 + n_{a6eb}M_6) &= n_{a2eb}M_2 , \\
V_a &= n_{a1}M_1/\rho_1 + n_{a2}M_2/\rho_2 + n_{a6}M_6/\rho_6 .
\end{aligned}$$

which play the role of nonlinear equality constraints. Constant data are given for

$$\begin{aligned}
M_1 &= 0.11911 , \quad \rho_1 = 1095 , \quad T_{ref1} = 363.16 , \\
M_2 &= 0.07412 , \quad \rho_2 = 809 , \quad T_{ref2} = 363.16 , \\
M_3 &= 0.19323 , \quad \rho_3 = 1415 , \quad T_{ref4} = 363.16 , \\
M_4 &= 0.31234 , \quad \rho_4 = 1528 , \quad T_{g2} = 363.16 , \\
M_5 &= 0.35733 , \quad \rho_5 = 1451 , \quad R = 8.314 , \\
M_6 &= 0.07806 , \quad \rho_6 = 1101 .
\end{aligned}$$

Model parameters to be estimated and for which some initial guesses are available, are

$$\begin{aligned}
k_{ref1} &= 5 \cdot 10^{-4} , \quad E_{a1} = 3.52 \cdot 10^4 , \\
k_{ref2} &= 8 \cdot 10^{-8} , \quad E_{a2} = 8.5 \cdot 10^4 , \\
k_{ref4} &= 1 \cdot 10^{-8} , \quad E_{a4} = 3.5 \cdot 10^4 , \\
k_{c2} &= 1.7 \cdot 10^{-1} , \quad d_{h2} = 1.08 .
\end{aligned}$$

The two input feed controls and the time-dependent temperature are piecewise linear functions defined at 10 grid points between $t = 8$ and $t = 80$. The corresponding 30 support values are experimental design parameters together with the bounded parameters MV_1 , MV_2 , MV_3 , g_a , g_{aea} , g_{aeb} , V_a and the parameters n_{a1} , n_{a2} , n_{a6} , n_{a1ea} , n_{a2eb} , n_{a6ea} , and n_{a6eb} , which are coupled by a set of seven equations mentioned above. To sum up, the whole optimization problem consists of 8 model parameters, 47 design parameters, 8 nonlinear equality constraints, and 20 linear inequality constraints to satisfy monotonicity of the input feeds. In addition, there are 10 time values between 0 and 80, and four measurement functions n_1 , n_3 , n_4 , and n_5 , and model variables are scaled to one.

The initial design is based on the data of Table 3.7. First, we suppose that the parameters given above, are the result of 'real' data fitting run. Experimental data are generated at the 10

time values and random errors based on a uniform distribution with relative deviation of 1 %. Subsequently, confidence intervals subject to model parameters are computed as described in Section 3.1, see (3.6). The results are listed in Table 3.8. The maximum standard deviation is more than 800 %, i.e., it is practically impossible to estimate the model parameters based on the given design data.

The code MODFIT is executed with termination tolerance 10^{-8} and $\epsilon = 10^{-2}$ for the approximation of partial derivatives. The optimization routine NLPQLP of Schittkowski [440] terminates after 72 iterations after reducing the performance criterion from 2.7×10^9 to $6.9 \cdot 10^4$. Optimal design values are listed in Table 3.7. Corresponding state and control functions are shown in Figures 3.6 to 3.9. As for the initial design, confidence intervals are computed for the design parameters, see Table 3.8. Now all deviations are below 15 %.

Table 3.7: Design Parameters before and after Experimental Design

p	initial	final
MV_1	1.0	0.24299
MV_2	0.3	1.13686
MV_3	0.3	0.25613
g_a	0.75	0.80000
g_{aea}	0.5	0.67401
g_{aeb}	0.4	0.32820
V_a	2.75	11.29776
n_{a1}	0.106	0.68736
n_{a2}	0.106	0.18089
n_{a6}	0.0876	0.30515
n_{a1ea}	0.0319	0.78145
n_{a2eb}	0.0319	0.17605
n_{a6ea}	0.0486	0.57685
n_{a6eb}	0.0454	0.34219

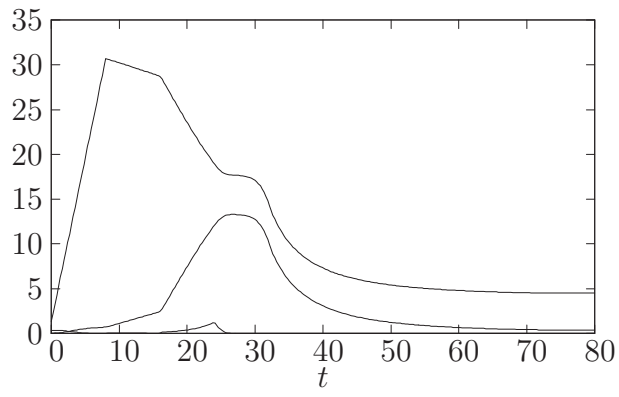


Figure 3.6: State Functions $n_1(t)$, $n_2(t)$, $n_3(t)$

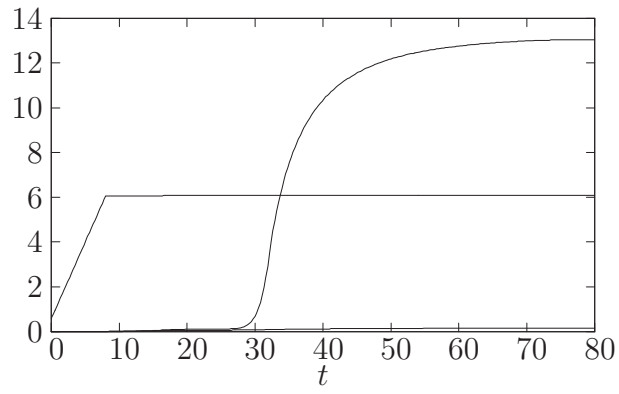


Figure 3.7: State Functions $n_4(t)$, $n_5(t)$, $n_6(t)$

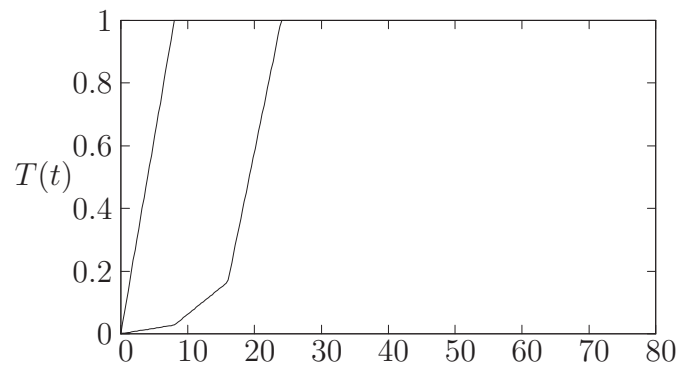


Figure 3.8: Control Functions $feed_a(t)$ and $feed_b(t)$

Table 3.8: Confidence Intervals for Urethane Problem before and after Experimental Design

p	initial (%)	final (%)	weights (%)
k_{ref1}	1.14	14.70	7.44
k_{ref2}	56.45	11.53	2.73
k_{ref4}	1.62	1.30	0.20
k_{c2}	859.13	4.71	0.00058
E_{a1}	0.77	0.18	2.34
E_{a2}	1.24	1.95	0.95
E_{a4}	1.38	0.84	0.14
d_{h2}	676.79	6.21	0.00026

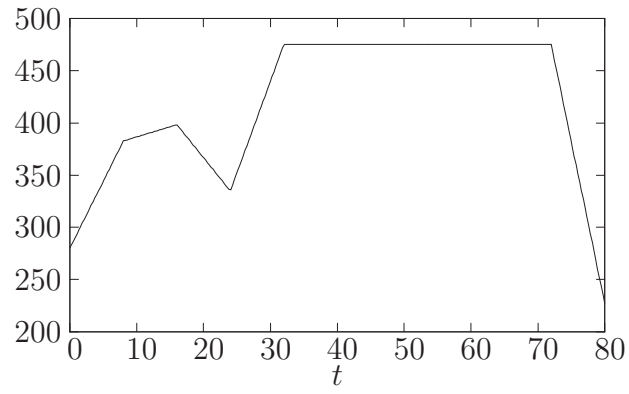


Figure 3.9: Temperature $T(t)$

3.4 Experimental Design with Weights

The experimental design approach introduced in the previous section assumes that the time values are known in advance. However, there are very many situations where one would like to know in advance their approximate number and also their optimal locations, to improve the confidence intervals of the parameters to be estimated, and to reduce the number of time-consuming or expensive experiments.

Our idea is to proceed from a given set of time values which could be large and dense, and to formulate an experimental design optimization problem as before by introducing additional weights w_k , $k = 1, \dots, l$. Thus, we replace the model function $h(p, q, t_k)$ by $w_k h(p, q, t_k)$ with additional weight factors w_k , $k = 1, \dots, l$, which are to be treated as optimization variables in our optimum design problem (3.15) and which becomes

$$\begin{aligned}
 & \min \text{trace}(C(w, p, q)) \\
 & g_j(p, q) = 0 \quad , \quad j = 1, \dots, m_e \quad , \\
 & g_j(p, q) = 0 \quad , \quad j = m_e + 1, \dots, m \quad , \\
 & q \in \mathbb{R}^{n_q}, w \in \mathbb{R}^l : \quad \sum_{k=1}^l w_k = 1 \quad , \\
 & q_l \leq q \leq q_u \quad , \\
 & \tau \leq w_k \leq 1 \quad , \quad k = 1, \dots, l \quad ,
 \end{aligned} \tag{3.22}$$

with covariance matrix $C(w, p, q) = I(w, p, q)^{-1}$ depending now on additional weights, $I(w, p, q) = F(w, p, q)F(w, p, q)^T$, $F(w, p, q) = \nabla_p f(w, p, q)$, and finally

$$f(w, p, q) = (w_1 h(p, q, t_1), \dots, w_l h(p, q, t_l))^T \quad .$$

Note that for stability reasons, a small lower bound τ is introduced for the weights.

Corresponding partial derivatives of the objective function

$$\phi(w, q) := \text{trace}(C(w, p, q))$$

subject to a weight w_k are obtained from

$$\begin{aligned}
 \frac{\partial}{\partial w_k} \phi(w, q) = & -\text{trace} \left(I(w, p, q)^{-1} \left(\frac{\partial}{\partial w_k} F(w, p, q) F(w, p, q)^T \right. \right. \\
 & \left. \left. + F(w, p, q) \frac{\partial}{\partial w_k} F(w, p, q)^T \right) I(w, p, q)^{-1} \right) \quad .
 \end{aligned} \tag{3.23}$$

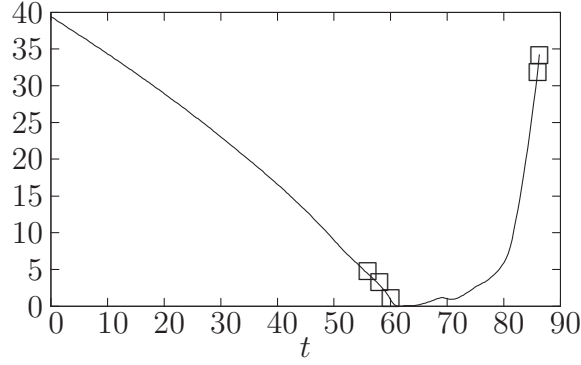


Figure 3.10: State Function $C_S(t)$

see (3.16), and from

$$\begin{aligned}
\frac{\partial}{\partial w_k} F(w, p, q) &= \frac{\partial}{\partial w_k} \nabla_p f(w, p, q) \\
&= \left(\frac{\partial^2}{\partial w_k \partial p_i} (w_k h(p, q, t_k)) \right)_{i=1, n_p; k=1, l} \\
&= \left(\frac{\partial}{\partial p_i} h(p, q, t_k) \right)_{i=1, n_p; k=1, l},
\end{aligned} \tag{3.24}$$

confer also (3.17). Thus, we get the weight derivatives more or less for free, since the partial derivatives subject to the model parameters are known from the computation of the objective function.

Example 3.6 (MICGROWX/Y/Z) We consider again Example 3.4, see Banga et al. [22]. In addition to the model and design parameters also weights are to be computed at an equidistant grid of 43 time values. Thus, the optimization problem (3.22) gets 86 additional variables. NLPQLP computes a solution in seven iterations with termination accuracy 10^{-6} . The total number of experiments is reduced to 5, see Figure 3.10, and would have to be taken into account only for C_S . Model parameter $K_p = 10$ is estimated subject to a confidence level 0.16 and $K_i = 0.1$ subject to 0.00011. Input feed is very similar to the optimal feed of Example 3.4. The location of the time values seem to be exactly at the critical points which determine the structure of the dynamical system.

Example 3.7 (URETHAN1/2) We consider again the urethane problem of Example 3.5 because of its practical relevance, see also (3.21). It is pointed out in Bauer et al. [29] that the experiments are expensive and that it is highly desirable to reduce their number as much as possible. We proceed from 40 equidistant time values between 0 and 80 for the four

Table 3.9: Optimal Weights for Urethane Problem

i	t_i	$n_1(t_i)$	$n_3(t_i)$	$n_4(t_i)$	$n_5(t_i)$
1	12	0.183	0.267		
2	16	0.0604			
3	18	0.0073			
4	20			0.0022	0.0018
5	26			0.0929	
6	28	0.03	0.131		0.0111
7	30				0.171
8	32	0.074			0.036
9	34				0.0035
10	36				
11	38	0.0123		0.0934	0.0025
12	40				0.0031
13	80	0.0068			

measurable output functions n_1 , n_3 , n_4 , and n_5 , and try to reduce their number to only the significant ones without losing the desired identification option as computed in the previous section. It is to be noted that all substrates are measured independently of each other, i.e., we have a total of $l = 160$ experimental data from where relevant ones are to be extracted. The optimization routine needs 132 iterations to reduce the performance criterion from $1.65 \cdot 10^{19}$ to $8.0 \cdot 10^{14}$ under the stopping tolerance 10^{-8} . Nineteen weights are above the lower bounds as shown in Table 3.9, and the corresponding confidence levels are found in Table 3.8. They are significantly smaller than in case of the 40 measurements taken in the previous section, and also the reduction of experimental expenses is significant.

Chapter 4

Numerical Algorithms

EASY-FIT^{*ModelDesign*} serves as a user interface for the parameter estimation programs MODFIT and PDEFIT that are also executable outside of **EASY-FIT**^{*ModelDesign*}. One of its features is the automatic generation of input files in ASCII format for the codes mentioned above. Model functions are either defined symbolically to be executed by the automatic differentiation tool PCOMP, or must be given in form of Fortran codes.

The corresponding data and code organization is documented in Chapter 8. In this chapter we describe very briefly the underlying numerical algorithms implemented.

4.1 Data Fitting Algorithms

The parameter estimation programs contain interfaces for a nonlinear least squares algorithm called NLPLSQ, see Schittkowski [446]. By transforming the original problem into a general nonlinear programming problem in a special way, typical features of a Gauss-Newton and quasi-Newton least squares method are retained, see Schittkowski [429] for details.

In case of least squares data fitting with very many measurements, the sum of squared functions is directly minimized by the SQP code NLPQLP [440]. The total number of iterations might increase, but the calculation time per iteration is decreased.

When minimizing a sum of absolute function values, i.e., the L_1 norm, the problem is transformed into a smooth nonlinear programming problem by introducing $2l$ additional variables and inequality constraints. The code is called NLPL1.

L_∞ -problems, where the maximum of absolute residual values is to be minimized, are solved by the code NLPINF [448]. One additional variable and l additional inequality constraints are introduced to transform the min-max problem into a smooth nonlinear optimization problem, which is then solved by NLPQLP [440]. In case of very many measurements, the transformed problem is solved by the active set code NLPQLB [442, 443].

4.2 Steady State Systems

The program MODFIT is executed to solve parameter estimation problems based on dynamical equations or steady state systems, respectively. To solve the corresponding systems of nonlinear equations, they are treated as a general nonlinear programming problems and solved by the Fortran code NLPQLP, see Schittkowski [427, 440, 449]. Objective function is the sum of squares of the system parameters, and the constraints are identical to the nonlinear system of equations given.

The algorithm proceeds from a successive quadratic approximation of the Lagrangian function and linearization of constraints. To get a search direction, a quadratic programming problem must be solved in each iteration step. A subsequent line search stabilizes the algorithm.

Also the starting values required to initialize an optimization cycle, must be predetermined by the user in a suitable way. They may depend on the parameters of the outer optimization problem. The system of nonlinear equations must be solved for each experimental time and concentration value. Moreover, the gradients of the model function $\bar{h}(p, z(p, t), t)$ are calculated analytically by the implicit function theorem. In this case, a system of linear equations must be solved for each time value by numerically stable Householder transformations.

4.3 Laplace Back-Transformation

MODFIT is also executed to solve parameter estimation problems, where the model functions are defined in the Laplace space. In this case, constraints are not allowed. Model functions and gradients are either declared in form of Fortran code or through the automatic differentiation features of the PCOMP language.

If an analytical back-transformation is not available, we have to apply a numerical quadrature formula, see Bellman et al. [33] for details. In our case, we use the quadrature formula of Stehfest [490]. Proceeding from a given Laplace transform $H(p, s, c) = L(h, p, s, c)$ of an unknown function $h(p, t, c)$, we compute the coefficients

$$v_i = (-1)^{q+i} \sum_{k=(i+1)/2}^{\min(i,q)} \frac{k^{q+1}(2k)!}{(q-k)!k!(k-1)!(i-k)!(2k-i)!} \quad (4.1)$$

which are independent of H and which can be evaluated before starting the main procedure, where function values are to be computed. Then

$$h(p, t, c) = \frac{\ln 2}{t} \sum_{i=1}^{2q} v_i H(p, \frac{i \ln 2}{t}, c) \quad (4.2)$$

is a numerical approximation formula for h .

The parameter for controlling the accuracy, is the number q . When working with double precision arithmetic, it is recommended to use $q = 5$ or $q = 6$. Any smaller value decreases the required accuracy, any larger value introduces additional round-off errors. For the practical models we have in mind, the numerical instabilities induced by oscillating function values, do not appear.

A particular advantage of the formula is that the derivative of h with respect to the parameters to be estimated, are easily obtained from the derivatives of H .

4.4 Ordinary Differential Equations

The parameter estimation program MODFIT that is executed by **EASY-FIT**^{*ModelDesign*} as an external executable file through the shell-feature of the Microsoft Visual Basic language, organizes data and evaluates the fitting functions with additional features, for example to process input data in a special format, to provide problem dependent output, or to generate plot information. The underlying dynamical model consists of a system of ordinary differential equations with initial values.

In case of parameter estimation in ordinary differential equations, it is possible to select either an implicit solver for stiff equations, RADAU5¹, an explicit solver for non-stiff equations, DOPRI5², or an explicit solver with internal numerical differentiation for non-stiff equations, see Benecke [36]. All codes apply Runge-Kutta method of order 4 to 5, the last one with additional sensitivity analysis for the evaluation of derivatives. For more details see Hairer, Nørsett and Wanner [197] or Hairer and Wanner [199], respectively.

Arbitrary linear or nonlinear constraints can be taken into account. For implicit methods, gradients of the right-hand side of the differential equation can be evaluated analytically using either user-provided derivatives or automatic differentiation.

The implicit code uses dense output, i.e., the integration is performed over the whole interval given by first and last time value, and intermediate solution values are interpolated. In this case, gradients with respect to the parameters to be estimated, are obtained by external numerical differentiation.

The explicit algorithm is capable to evaluate derivatives of the solution of the ODE internally with respect to the parameters to be estimated, i.e., by analytical differentiation of the Runge-Kutta scheme.

It is possible that the right-hand side of an ODE is non-continuous subject to integration time, for example if non-continuous input functions exist. Especially in case of short peaks, the integration routine might not realize the peak at all because of a big time step. Moreover, the numerical approximation of gradients could become unstable in case of discontinuities. Thus, MODFIT allows to supply an optional number time values, so-called break or switching points, where the integration of the ODE is restarted with initial tolerances, for example with the initially given stepsize. The integration in the proceeding interval is stopped at the time value given minus a relative error in the order of the machine precision. Note also that break points can be treated as optimization variables, i.e., may vary from one iteration step to the other.

¹Copyright ©2004, Ernst Hairer

²Copyright ©2004, Ernst Hairer

4.5 Differential Algebraic Equations

In this situation, **EASY-FIT**^{*ModelDesign*} calls again the parameter estimation program MODFIT as an external executable file, where the underlying model is given by a system of differential algebraic equations. Gradients of the right-hand side of the differential equation with respect to system variables are evaluated analytically using either user-provided derivatives or automatic differentiation. The algebraic differential equation is solved by an implicit Runge-Kutta code of Radau-type, RADAU5³, confer Hairer and Wanner [199]. DAE's with an index up to three can be integrated.

If consistent initial values cannot be provided by the user, the corresponding nonlinear system of equations is treated as general nonlinear programming problem with equality constraints. A minimum norm solution is computed by the sequential quadratic programming method NLPQLP of Schittkowski [427, 440, 449]. The initial values given for the algebraic equations are used as starting values.

For reasons outlined in the previous section, it is possible that the right-hand side of an DAE becomes non-continuous with respect to integration time. Thus, it is possible to supply an optional number time values, where the integration of the DAE is restarted with initial tolerances, for example with the initially given stepsize. The integration in the proceeding interval is stopped at the time value given minus a relative error in the order of the machine precision. Break or switching points are either constant or optimization variables to be adapted by the optimization code.

³Copyright ©2004, Ernst Hairer

4.6 Partial Differential Equations

The underlying idea is to transform the partial differential equations into a system of ordinary differential equations by discretizing the model functions subject to the spatial variable x . This approach is known as the method of lines, see Schiesser [420].

For the i -th integration interval of the spatial variable, we denote the number of discretization points by n_i , $i = 1, \dots, n_t$. We proceed from uniform grid points within each interval and get a discretization of the whole space interval from x_L to x_R . To approximate the first and second partial derivatives of $u^i(p, x, t)$ with respect to the spatial variable at a given point x , several different alternatives have been implemented in PDEFIT:

a) Difference Formulae: First and second derivatives can be approximated by difference formulae, see Schiesser [420]. Difference formulae with 3 and 5 points for first derivatives are available, that can be applied recursively to also get the second derivatives. Alternatively a 5-point difference formula for second derivatives is implemented as well. The difference formulae are adapted at the boundary to accept given function and gradient values. Moreover, first derivatives can be approximated by simple forward and backward differences. These formulae are recommended if there are steep fronts, for example in case of transportation or fluid dynamics, where symmetric procedures lead to numerical irregularities. To apply one of these differences, the flow direction must be known in advance. They are particularly useful in a situation, where an upwind formula is desirable, but the right-hand side of the PDE is not given in flux form, see below. Most of these difference formulae can be combined and applied individually to the spatial derivatives of the state variables under consideration.

b) Upwind Formulae for Hyperbolic Equations: In case of a scalar hyperbolic equation

$$u_t^i = f^i(p, u^i, x, t)_x \quad , \quad (4.3)$$

$i = 1, \dots, n_t$, with a so-called flux function f , approximation by difference formulae might become unstable especially if non-continuous boundary conditions are supplied to describe the propagation of shocks, see Schiesser [420] for some numerical examples. The following upwind formulae are available for solving hyperbolic equations:

- simple upwind formula
- second order TVD-scheme
- third order upwind-biased TVD-scheme

For more information, see the original literature, e.g. Yee [557], Chakravarthy and Osher [82], [83], [84], Sweby [500], Wang and Richards [541], and Yang and Przekwas [556]. TVD stands for total variation diminishing and the corresponding one parameter family of

upwind formulae was proposed by Chakravarthy and Osher [82]. In this case, a certain stability criterion requires that the internal time stepsizes of the ODE-solver do not become too small compared to the spatial discretization accuracy. Because of the black box approach used, the stepsizes, however, cannot be modified and we have to suppose that the criterion remains satisfied.

c) ENO-Method for Systems of Advection Equations: Systems of non-homogeneous, nonlinear advection equations

$$u_t^i = f_1^i(p, u^i)_x + f_2^i(p, u^i, u_x^i, x, t) \quad (4.4)$$

with area index i , $i = 1, \dots, n_t$, with $u^i \in \mathbb{R}^{n_p}$, $n_p \geq 1$, can be solved by essentially non-oscillatory (ENO) schemes, see Harten, Engquist, Osher, and Chakravarthy [206], Harten [207], or Walsteijn [536]. High order polynomials are applied to approximate a so-called primitive function, which is supposed to represent the flux function at intermediate spatial grid points. The choice of the corresponding stencil depends on the magnitude of divided differences, to direct the stencil away from discontinuities. To solve also systems of hyperbolic equations, a full eigenvalue-eigenvector decomposition of the Jacobian of the flux function is performed with respect to u^i , and the scalar ENO method is applied to coefficient functions after a suitable transformation. Flux splitting at the cell walls is applied, requiring separate decompositions for the left and right approximation, see Donat and Marquina [120] and Marquina and Donat [321]. The wind direction is estimated by the corresponding eigenvalue. The resulting system of ordinary differential equations can be solved either by an implicit or explicit ODE solver as before, or by a special Runge-Kutta method with fixed stepsizes to satisfy the CLF condition.

Whenever a boundary condition in Dirichlet-form

$$\begin{aligned} u_k^1(p, x_L, t) &= u_k^L(p, t) \\ u_k^{n_t}(p, x_R, t) &= u_k^R(p, t) \\ u_k^i(p, x_i^a, t) &= c_{i,k}^R(p, u^{i+1}(p, x_i^a, t), t) \\ u_k^i(p, x_{i-1}^a, t) &= c_{i-1,k}^L(p, u^{i-1}(p, x_{i-1}^a, t), t) \end{aligned} \quad (4.5)$$

is given for $1 \leq k \leq n_p$, then we know the value of the boundary function and use it to interpolate or approximate the function $u(p, x, t)$ as described above. In other words, the corresponding function value in the right-hand side of the discretized system is replaced by the value given.

Alternatively a boundary condition may appear in Neumann-form

$$\begin{aligned} u_{k,x}^1(p, x_L, t) &= \hat{u}_k^L(p, t) \\ u_{k,x}^{n_t}(p, x_R, t) &= \hat{u}_k^R(p, t) \\ u_{k,x}^i(p, x_i^a, t) &= \hat{c}_{i,k}^R(p, u^{i+1}(p, x_i^a, t), u_x^{i+1}(p, x_i^a, t), t) \\ u_{k,x}^i(p, x_{i-1}^a, t) &= \hat{c}_{i-1,k}^L(p, u^{i-1}(p, x_{i-1}^a, t), u_x^{i-1}(p, x_{i-1}^a, t), t) \end{aligned} \quad (4.6)$$

for $1 \leq k \leq n_p$. In this case, the derivative values at the boundary are replaced by the given ones before evaluating the second order spatial derivative approximations.

Ordinary differential equations are added to the discretized system without any further modification. Since arbitrary coupling points are allowed, they are rounded to the nearest line of the discretized system. In the same way, fitting criteria can be defined at arbitrary values of the spatial variable.

When defining the transition function, it is important to have the underlying flux direction in mind. If, for example, the flux is in the direction of the spatial variable and we want to define a continuous transition at x_i^a , then we have to formulate the corresponding transition function in the form $u_k^{i+1}(p, x_i^a, t) = u_k^i(p, x_i^a, t)$ in order to guarantee that the boundary values at x_L are spread over the interval.

For the same reasons outlined in the previous sections, it is possible that the right-hand side of a PDE becomes non-continuous with respect to integration time. Thus, it is possible to supply an optional number time values, where the integration of the DAE is restarted with initial tolerances. The integration in the proceeding interval is stopped at the time value given minus a relative error in the order of the machine precision. Break or switching points are either constant or optimization variables to be adapted by the optimization code.

In many application models, we need to compute an integral with respect to the spatial variable x for example to evaluate a mass balance,

$$\int_{x_{j-1}^a}^{x_j^a} u^i(p, x, t) dx$$

where the integral is taken over the j -th area where the PDE is defined, $j = 1, \dots, n_t$, and where $i = 1, \dots, n_p$. The integral is evaluated by Simpson's rule and can be retrieved from a common block or, alternatively, through a special construct of the PCOMP language.

4.7 Partial Differential Algebraic Equations

The basic idea is now to transform the partial differential into a system of differential algebraic equations by discretizing the model functions with respect to the spatial variable x . Again, we denote the number of discretization points by n_i , $i = 1, \dots, n_t$, for the i -th integration interval of the spatial variable. We proceed from uniform grid points within each interval and get a discretization of the whole space interval from x_L to x_R . To approximate the first and second partial derivatives of $u^i(p, x, t)$ with respect to the spatial variable at a given point x , we may apply any difference formula as outlined in the previous section. Thus, we get a system of differential algebraic equations that can be solved then by any of the available integration routines.

Boundary conditions have to satisfy the algebraic equations. Consistent initial values are computed within the code PDEFIT, where the given data serve as starting parameters for the nonlinear programming algorithm applied. Consequently, we allow only index-1-systems unless it is guaranteed that consistent initial values for the discretized DAE are available. Also any jumps or discontinuities at initial values of algebraic equations do not make sense.

4.8 Statistical Analysis

Proceeding from the assumption that the model is sufficiently linear in a neighborhood of an optimal solution vector and that all experimental data are Gaussian and independent, some statistical data can be evaluated:

- Variance/covariance matrix of the problem data
- Correlation matrix of the problem data
- Estimated variance of residuals
- Confidence intervals for the individual parameters subject to the significance levels 1%, 5% or 10%, respectively.

A priority analysis is performed after a data fitting run. If a model seems to be overdetermined, the computed levels give an impression of the significance of parameters and help to decide upon questions like

- which parameters can be identified,
- which parameters should be kept fixed,
- whether additional experimental data must be required.

Moreover, overdetermined data fitting problems often lead to unstable and slow convergence of Gauss-Newton-type least squares algorithms.

To detect the significant parameters on the one hand and the redundant or dependent parameters on the other we apply the following procedure. Successively parameters are eliminated from the Fisher information matrix until one of the following conditions is satisfied:

1. The smallest eigenvalue of the Fisher information matrix is smaller than a threshold value, i.e. the given significance level.
2. The maximum parameter correlations are significantly reduced (25 %).
3. None of the above termination reasons are met and all parameters have been eliminated.

Level 1 corresponds to the first eliminated variables, level 2 to the second, etc. The final level can be assigned to several parameters indicating a group of identifiable parameters. Eigenvalues and eigenvectors are computed by subroutine DSPEV of the LAPACK⁴ library [7].

⁴Copyright ©1992-2007, The University of Tennessee

Experimental design depends on two steps. First, we have to setup the covariance matrix and estimate its volume, in our case by the diagonal elements or confidence intervals, respectively. Subsequently, the nonlinear programming code NLPQLP of Schittkowski [440, 440, 449] is started, see also Schittkowski [427]. Since the sequential quadratic programming code requires the calculation of gradients of the objective function and all constraints, the procedure outlined in Section 3.3 is applied. It is known that the Fisher information matrix is often only semi-definite and rank-deficient, especially in case of additional equality constraints. Thus, the generalized inverse is computed by the LAPACK routine DGELSS based on a singular value decomposition. The tolerance for detecting the rank is set to the same tolerance by which the quadratic programming solver of NLPQLP is called, i.e., to 10^{-12} .

It must be emphasized that the choice of a tolerance for the approximation of mixed partial derivatives by forward differences has a crucial impact on the performance of the algorithm, and should be adapted carefully. Internally, objective function values are scaled by the starting value, i.e., the initial design.

Chapter 5

The Modeling Language PCOMP

Within the user-interface of **EASY-FIT**^{*ModelDesign*}, the numerical algorithms are implemented in a way that the nonlinear model functions defining fitting criteria, dynamical model equations and constraints, are evaluated either by a user provided Fortran code or by the interpreter PCOMP. In the first case, one has to code the model function subject to a frame that is inserted in the editor when generating a new problem. The usage is completely described by initial comments and is not repeated here.

In the second case, all data, variables and functions defining the model functions, must be written on a text file in a format similar to Fortran, are pre-compiled internally before starting the optimization cycle. Proceeding from the intermediate code generated, function and gradient values are evaluated from the code during run time. A particular advantage is that gradients, as far as needed, are calculated automatically without any numerical approximation errors.

5.1 Automatic Differentiation

Let $f(p)$ be a nonlinear differentiable function with real values defined for all $p \in \mathbb{R}^n$. By automatic differentiation we understand the numerical computation of a derivative value $\nabla f(p)$ of f at a given point p without truncation errors and without hand-coded formulas.

Numerical differentiation requires at least n additional function evaluations for one gradient calculation and induces truncation errors. Although very easy to implement, the numerical errors are often not tolerable, especially when the derivatives are used within another numerical approximation scheme. A typical example is the differentiation of solutions of unstable differential equations in a parameter estimation problem.

Automatic differentiation overcomes the drawbacks mentioned and is a very useful tool in all practical applications that require derivatives. The resulting code can be used for the evaluation of nonlinear function values by interpreting symbolic function input without extra compilation and linking. Whenever needed, gradients can be evaluated exactly at run time.

There exists meanwhile a large variety of different computer codes for automatic differentiation, see Juedes [243] for a review. They differ in the underlying design strategy, domain of application, mathematical method, implementation and numerical performance. The code PCOMP that is to be introduced now, is based on a somewhat restricted language related to Fortran, but with emphasis on code flexibility and speed.

Basically there are two ways to implement automatic differentiation, called forward and backward accumulation respectively. Both are used in PCOMP, one for the direct evaluation of function and constraints values, the other one for generation of Fortran code, see Dobmann, Liepelt and Schittkowski [115] for details. The first variant was implemented for the parameter estimation codes we are interested in.

Note that a particular advantage of gradient calculations in reverse accumulation mode is the limitation of relative numerical effort by a constant that is independent of the dimension, i.e., the number of variables. A review of further literature and a more extensive discussion of symbolic and automatic differentiation is given in Griewank [186]. An up-to-date summary of related papers is published in Griewank and Corliss [184].

First we have to investigate the question how a nonlinear function is evaluated. The idea is to break a given expression into elementary operations that can be evaluated either by internal compiler operations directly or by external function calls. For a given function f the existence of a sequence f_i of elementary functions is assumed, where each individual function $\{f_i\}$ is real-valued and defined on \mathbb{R}^{n_i} , $1 \leq n_i \leq m - 1$ for $i = n + 1, \dots, m$. We define now the situation more formally by a pseudo-program.

Definition 5.1 *Let f be a real-valued function defined on the \mathbb{R}^n . Then some real valued functions f_i defined on \mathbb{R}^{n_i} , $i = n + 1, \dots, m$, are called a sequence of elementary functions for f , $m \geq n$, if there exists an index set J_i with $J_i \subset \{1, \dots, i-1\}$, $|J_i| = n_i$ for each function f_i , $i = n + 1, \dots, m$, such that any function value of f for a given vector $p = (p_1, \dots, p_n)^T$ can be evaluated according to the following program:*

$$\begin{aligned} &\text{For } i = n + 1, \dots, m \text{ let} \\ &\quad p_i = f_i(p_k, k \in J_i) \\ &\text{Let } f(p) = p_m \end{aligned} \tag{5.1}$$

The proposed way of evaluating function values is implemented in any compiler or interpreter of a higher programming language, if we omit possible code optimization considerations. In computer science terminology, we would say that a postfix expression is built in the form of a stack, which is then evaluated recursively. Thus, the elementary functions can be obtained very easily and the corresponding technique is found in any introductory computer science textbook.

Note that for every function $f(p)$ there exists at least one trivial sequence of elementary functions by $m = n + 1$ and $f_{n+1}(p) = f(p)$. For practical use, however, we assume that the functions f_i are basic machine operations, intrinsic or external functions, where the relative

evaluation effort is limited by a constant independently of n . Under this condition, suitable bounds for the work ratio can be proved. The algorithm can be implemented efficiently by using stack operations, which reduce the storage requirements as far as possible, i.e., we do not need to store all intermediate variables p_{n+1}, \dots, p_m .

By investigation of the above program for evaluating a function value $f(p)$, we realize immediately that in a very straightforward way the gradient $\nabla f(p)$ can be evaluated simultaneously. If we know how the derivatives of the elementary functions can be obtained, the only thing we have to change is the inclusion of another program line for the gradient update by exploiting the chain rule. In a natural way we denote the resulting approach as *forward accumulation*.

Definition 5.2 *Let f be a differentiable function and $\{f_i\}$ be a sequence of elementary functions for evaluating f with corresponding index sets $J_i, i = n + 1, \dots, m$. Then the gradient $\nabla f(p)$ for a given $p \in \mathbb{R}^n$ is determined by the following program:*

$$\begin{aligned}
&\text{For } i = 1, \dots, n \text{ let} \\
&\quad \nabla p_i = e_i \\
&\text{For } i = n + 1, \dots, m \text{ let} \\
&\quad p_i = f_i(p_k, k \in J_i), \\
&\quad \nabla p_i = \sum_{j \in J_i} \frac{\partial f_i(p_k, k \in J_i)}{\partial p_j} \nabla p_j \\
&\text{Let } f(p) = p_m, \\
&\quad \nabla f(p) = \nabla p_m
\end{aligned} \tag{5.2}$$

Here e_i denotes the i -th axis vector in $\mathbb{R}^n, i = 1, \dots, n$. Again the evaluation of gradients can be performed by suitable stack operations, reducing the memory requirements.

The complexity of the forward accumulation algorithm is bounded by a constant times n , the number of variables. In other words, the numerical work is the same order of magnitude as for numerical differentiation.

5.2 Input Format for PCOMP

The symbolic input of nonlinear functions is only possible if certain syntax rules are satisfied. The PCOMP-language is a subset of Fortran with a few extensions, see Dobmann, Liepelt, Schittkowski and Trassl [116] for details. In particular, the declaration and executable statements must satisfy the usual Fortran input format, i.e., must start at column 7 or subsequently. A statement line is read in until column 72. Comments beginning with `C` at the first column, may be included in a program text wherever needed. Statements may be continued on subsequent lines by including a continuation mark in the 6th column. Either capital or small letters are allowed for identifiers of the user and key words of the language. The length of an identifier has to be smaller than 20 tokens.

In contrast to Fortran, however, most variables are declared implicitly by their assignment statements. Variables and functions must be declared separately only if they are used for automatic differentiation. PCOMP possesses special constructs to identify program blocks.

- * **PARAMETER**

Declaration of constant integer parameters to be used throughout the program, particularly for dimensioning index sets.

- * **SET OF INDICES**

Definition of index sets that can be used to declare data, variables and functions or to define `sum` or `prod` statements.

- * **INDEX**

Definition of an index variable, which can be used in a **FUNCTION** program block.

- * **REAL CONSTANT**

Definition of real data, either without index or with one- or two-dimensional index. An index may be a variable or a constant number within an index set. Also arithmetic expressions may be included.

- * **INTEGER CONSTANT**

Definition of integer data, either without index or with one- or two-dimensional index. An index may be a variable or a constant number within an index set. Also arithmetic integer expressions may be included.

- * **TABLE <identifier>**

Assignment of constant real numbers to one- or two-dimensional array elements. In subsequent lines, one has to specify one or two indices followed by one real value per line in a free format (starting at column 7 or later).

- * **VARIABLE**

Declaration of variables with up to one index, with respect to which automatic differentiation is to be performed.

- * **CONINT** <identifier>
Declaration of a piecewise constant interpolation function.
- * **LININT** <identifier>
Declaration of a piecewise linear interpolation function.
- * **SPLINE** <identifier>
Declaration of a spline interpolation function.
- * **MACRO** <identifier>
Definition of a macro function, an arbitrary set of PCOMP statements that define an auxiliary function to be inserted into subsequent function declaration blocks. Macros are identified by a name that can be used in any right-hand side of an assignment statement.
- * **FUNCTION** <identifier>
Declaration of functions either with up to one index, for which function and derivative values are to be evaluated. The subsequent statements must assign a numerical value to the function identifier.
- * **END**
End of the program.

It is recommended to follow the order of the above program blocks. They may be repeated whenever desirable. Data must be defined before their usage in a subsequent block. All lines after the final **END** statement are ignored by PCOMP. The statements within the program blocks are very similar to usual Fortran notation and must satisfy the following guidelines:

Constant data: For defining real numbers either in analytical expressions or within the special constant data definition block, the usual Fortran convention can be used. In particular the F-, E- or D-format is allowed.

Identifier names: Names of identifiers for variables and functions, index sets and constant data, must begin with a letter and the number of characters, i.e. letters, digits and underscores, must not exceed 20.

Index sets: Index sets are required for the **SUM** and **PROD** expressions and for defining indexed data, variables and functions. They can be defined in different ways:

1. Range of indices,

$$\text{ind1} = 1..27$$

2. Set of indices,

```
ind2 = 3,1,17,27,20
```

3. Computed index sets,

```
ind3 = 5*i + 100 , i=1..n
```

4. Parameterized index sets,

```
ind4 = n..m
```

Assignment statements: As in Fortran, assignment statements are used to assign a numerical value to an identifier, which may be either the name of the function that is to be defined, or of an auxiliary variable that is used in subsequent expressions,

```
r1 = p1*p4 + p2*p4 + p3*p2 - 11
r2 = p1 + 10*p2 - p3 + p4 + p2*p4*(p3 - p1)
f  = r1**2 + r2**2
```

Analytical expressions: An analytical expression is, as in Fortran, any allowed combination of constant data, identifiers, elementary or intrinsic arithmetic operations and the special SUM- and PROD-statements. Elementary operations are

+ , - , * , / , **

Note that PCOMP handles integer expressions in exponents in the same way as real expressions, and one should avoid non-positive arguments. Integer constants are treated as usual arithmetic operations. Allowed intrinsic functions are

```
ABS, SIN, COS, TAN, ASIN, ACOS, ATAN, SINH, COSH
TANH, ASINH, ACOSH, ATANH, EXP, LOG, LOG10, SQRT
```

Alternatively, the corresponding double precision Fortran names possessing an initial D can be used as well. Brackets are allowed to combine groups of operations. Possible expressions are

```
5*DEXP(-z(i))
```

or

```
LOG(1 + SQRT(c1*f1)**2)
```

INDEX–Variables: In PCOMP it is possible to define indices separately to avoid unnecessary differentiation of integer variables. They have to be defined in the program block INDEX,

```
*      INDEX
      i,j
      l
```

It is allowed to manipulate the index by statements of the form

```
i = 1+2*4-3
i = a(1)
f = a(i+2)+i*2.0
f = SUM(a(m-i), m IN ind)
f = i
f = g(i)
```

In this case, **a** must be declared in form of in integer array. However, the following assignment statements are not allowed, if **b** is a real array:

```
i = b(3)
i = 1.0
i = 4/2
f(i) = 3.0
```

Interpolation functions: PCOMP admits the interpolation of user defined data, using either a piecewise constant, piecewise linear, or a cubic spline function. Given n pairs of real values $(t_1, y_1), \dots, (t_n, y_n)$, we are looking for a nonlinear function interpolating these data. In the first case, we define a piecewise constant interpolation by

$$c(t) = \begin{cases} 0 & , \ t < t_1 \ , \\ y_i & , \ t_i \leq t < t_{i+1} \ , \ i = 1, \dots, n-1 \ , \\ y_n & , \ t_n \leq t \ . \end{cases}$$

A continuous piecewise linear interpolation function is

$$l(t) = \begin{cases} y_1 & , \quad t < t_1 \quad , \\ y_i + \frac{t - t_i}{t_{i+1} - t_i}(y_{i+1} - y_i) & , \quad t_i \leq t < t_{i+1} \quad , \quad i = 1, \dots, n-1 \quad , \\ y_n & , \quad t_n \leq t \quad , \end{cases}$$

and a cubic spline is given by

$$s(t) = \begin{cases} p(t; t_1, t_2, t_3, t_4, y_1, y_2, y_3, y_4) & , \quad t < t_4 \quad , \\ \bar{s}(t; t_4, \dots, t_n, y_4, \dots, y_n, \frac{d}{dt}p(t_4; \dots), 0) & , \quad t_4 \leq t \quad , \end{cases}$$

where $p(t; t_1, t_2, t_3, t_4, y_1, y_2, y_3, y_4)$ is a cubic polynomial with

$$p(t_i; t_1, t_2, t_3, t_4, y_1, y_2, y_3, y_4) = y_i \quad , \quad i = 1, \dots, 4 \quad ,$$

and $\bar{s}(t; \bar{t}_1, \dots, \bar{t}_m, \bar{y}_1, \dots, \bar{y}_m, \bar{y}'_1, \bar{y}'_m)$ a cubic spline function interpolating $(\bar{t}_1, \bar{y}_1), \dots, (\bar{t}_m, \bar{y}_m)$ subject to the boundary conditions

$$\frac{d}{dt}\bar{s}(\bar{t}_i; \bar{t}_1, \dots, \bar{t}_m, \bar{y}_1, \dots, \bar{y}_m, \bar{y}'_1, \bar{y}'_m) = \bar{y}'_i \quad , \quad i = 1 \text{ and } i = m \quad .$$

It is essential to understand that the constant and spline interpolation functions are not symmetric. Our main interest are dynamical systems, say ordinary or partial differential equations, where the initial value is set to 0 without loss of generality, leading to a non-symmetric domain. Moreover, interpolated data are often based on experiments that reach a steady state, i.e., a constant value. Thus, a zero derivative is chosen at the right end point for spline interpolation to facilitate the input of interpolated steady state data. On the other hand, any other boundary conditions can be enforced by adding artificial interpolation data.

The spline functions generated, are twice differentiable with the exception of the fourth break point. At this point, there exists only the first derivative and PCOMP generates the right-hand side differential quotient for the second derivative. We need at least four pairs of data to construct a spline interpolation as outlined above.

To given an example, we assume that we want to interpolate the nonlinear function $f(t)$ given by the discrete values $f(t_i) = y_i$ from Table 5.1, using the different techniques mentioned above.

Interpolation functions are defined by a program block starting with the keyword CONINT for piecewise constant functions, LININT for piecewise linear functions, or SPLINE for piecewise cubic splines, followed by the name of the function. The numerical values of the break points and the function values are given on the subsequent lines, using any standard format starting at column 7 or later. Using piecewise constant approximations, we get for our example:

i	t_i	y_i
1	0.0	0.00
2	1.0	4.91
3	2.0	4.43
4	3.0	3.57
5	4.0	2.80
6	5.0	2.19
7	6.0	1.73
8	7.0	1.39
9	8.0	1.16
10	9.0	1.04
11	10.0	1.00

Table 5.1: Interpolation data

```
*      CONINT F
      0.0  0.00
      1.0  4.91
      2.0  4.43
      3.0  3.57
      4.0  2.80
      5.0  2.19
      6.0  1.73
      7.0  1.39
      8.0  1.16
      9.0  1.04
     10.0  1.00
```

Within a function definition block, the interpolation functions are treated as intrinsic Fortran functions, that is, they have to contain a variable or constant as a parameter. If we assume that T has previously been declared as a variable, a valid statement could look like

```
*      FUNCTION OBJ
      OBJ = F(T)
```

The resulting approximations for piecewise constant functions, piecewise linear functions, or piecewise cubic spline functions are depicted in Figures 5.1–5.3. Whereas the cubic spline approximation is twice differentiable on the whole interval, the other two approximations are not differentiable at the break points and PCOMP uses the right-hand sided derivatives instead.

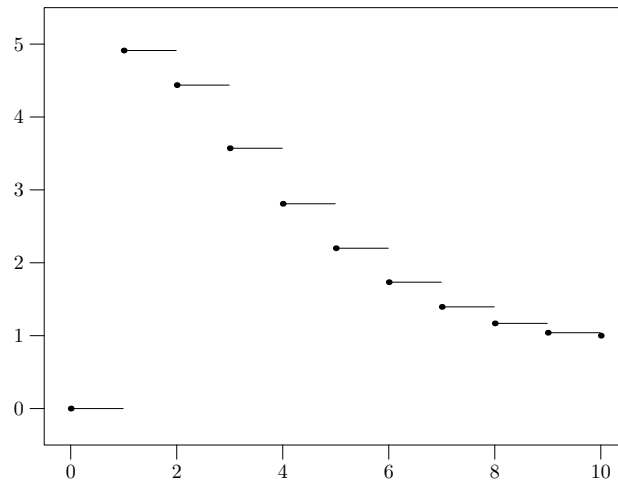


Figure 5.1: Piecewise Constant Interpolation

Macros: PCOMP does not allow the declaration of subroutines. However, it is possible to define macros, that is, arbitrary sequences of PCOMP statements that define an auxiliary variable to be inserted into the beginning of subsequent function declaration blocks. Macros are identified by a name that can be used in any right-hand side of an assignment statement

```
*      MACRO <identifier>
```

followed by a group of PCOMP statements that assign a numerical value to the given identifier. This group of statements is inserted into the source code block that contains the macro name. Macros have no arguments, but they may access all variables, constants, or functions that have been declared up to their first usage. Any values assigned to local variables within a macro, are also available outside in the corresponding function block.

If we assume that x is a variable and we want to define a macro that computes the square of x , we would write something like

```
*      MACRO sqr
      sqr = x*x
```

Now it is possible to replace each occurrence of the term $x*x$ with an invocation of the macro that we have just defined, for example

```
f = sqr-5.2
```

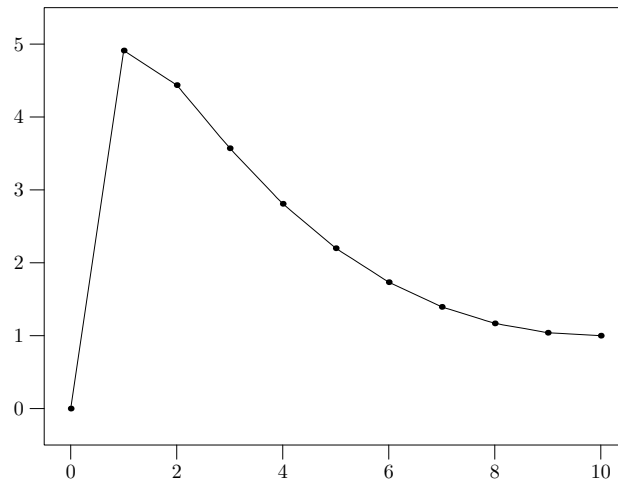



Figure 5.2: Piecewise Linear Interpolation

SUM– and PROD–expressions: Sums and products over predetermined index sets are formulated by SUM and PROD expressions, where the corresponding index and the index set must be specified, for example in the form

$$f = 100 * \text{PROD}(p(i) ** a(i), i \text{ IN } \text{inda})$$

In the above example, $p(i)$ might be a variable vector defined by an index set, and $a(i)$ an array of constant data.

Control statements: To control the execution of a program, the conditional statements

```
IF <condition> THEN
  <statements>
ENDIF
```

or

```
IF <condition> THEN
  <statements>
ELSE
  <statements>
ENDIF
```

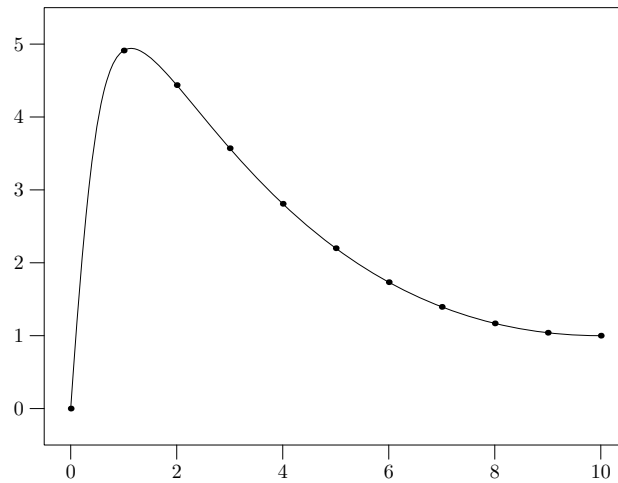


Figure 5.3: Piecewise Cubic Spline Interpolation

can be inserted into a program. Conditions are defined as in Fortran by the comparative operators `.EQ.`, `.NE.`, `.LE.`, `.LT.`, `.GE.`, `.GT.`, which can be combined using brackets and the logical operators `.AND.`, `.OR.` and `.NOT.`

The `GOTO`– and the `CONTINUE`–statements are further possibilities to control the execution of a program. The syntax for these statements is

```
GOTO <label>
```

and

```
<label> CONTINUE
```

where `label` has to be a number between 0 and 9999. Since PCOMP produces labels during the generation of the Fortran code in the reverse mode, it is advisable to use labels between 5000 and 9999. The `<label>` part of the `CONTINUE`–statement has to be located between columns 2 and 5 of an input line. Together with an index, the `GOTO`–statement can be used for example to simulate `DO`–loops, which are forbidden in PCOMP,

```

i = 1
s = 0.0
6000 CONTINUE
s = s + a(i)*b(i)
i = i+1

```

```

      IF (i.LE.n) THEN
        GOTO 6000
      ENDIF

```

Whenever indices are used within arithmetic expressions, it is allowed to insert polynomial expressions of indices from a given set. However, functions must be treated in a particular way. Since the design goal is to generate short, efficient Fortran codes, indexed function names can be used only in exactly the same way as defined. In other words, if a set of functions is declared by

```
* FUNCTION f(i), i IN index
```

then only an access to $f(i)$ is allowed, not to $f(1)$ or $f(j)$, for example. In other words, PCOMP does not extend the indexed functions to a sequence of single expressions similar to the treatment of SUM and PROD statements.

In PCOMP, it is allowed to pass variable values from one function block to the other. However, the user must be aware of a possible failure, if in the calling program the evaluation of a gradient value in the first block is skipped.

One should be very careful when using the conditional statement IF. Possible traps that prevent a correct differentiation are reported in Fischer [149], and are to be illustrated by an example. Consider the function $f(p) = p^2$ for $n = 1$. A syntactically correct formulation would be:

```

      IF (p.EQ.1) THEN
        f = 1
      ELSE
        f = p**2
      ENDIF

```

In this case, PCOMP would try to differentiate both branches of the conditional statement. If p is equal to 1, the derivative value of f is 0; otherwise it is $2p$. Obviously we get a wrong answer for $p = 1$. This is a basic drawback for all automatic differentiation algorithms of the type under consideration.

PCOMP allows the execution of external statements that must be linked to PCOMP in a special way, see Dobmann, Liepelt, Schittkowski and Trassl [116]. A frequently needed computational value in case of a PDE model is the integral with respect to the spatial variable x , i.e.,

$$\int_{x_{j-1}^a}^{x_j^a} u^i(p, x, t) dx$$

where the integral is taken over the j -th area where the PDE is defined, $j = 1, \dots, n_t$. Index i denotes the i -th solution component we want to integrate, $i = 1, \dots, n_p$. The integral is evaluated by Simpson's rule and denoted by

SIMPSN(I,J)

in the PCOMP language. This name can be inserted in an arithmetic expression, for example to compute a fitting criterion. The corresponding time value is either a measurement value or an intermediate value needed for generating plot data.

5.3 Error Messages of PCOMP

PCOMP reports error messages in the form of integer values of the variable IERR and, whenever possible, also line numbers LNUM. The meaning of the messages is listed in the following table. Note that the corresponding text is displayed if the error routine SYMERR is called with parameters LNUM and IERR.

In the version implemented for the parameter estimation codes, an error is reported when starting the execution of a numerical algorithm, i.e., when the parser analyzes the code. The corresponding error code and a line number are displayed and a user should edit the PCOMP code before trying it again.

- | | | |
|----|---|------------------------------------|
| 1 | - | file not found |
| 2 | - | file too long |
| 3 | - | identifier expected |
| 4 | - | multiple definition of identifier |
| 5 | - | comma expected |
| 6 | - | left bracket expected |
| 7 | - | identifier not declared |
| 8 | - | data types do not fit together |
| 9 | - | division by zero |
| 10 | - | constant expected |
| 11 | - | operator expected |
| 12 | - | unexpected end of file |
| 13 | - | range operator '..' expected |
| 14 | - | right bracket ')' expected |
| 15 | - | 'THEN' expected |
| 16 | - | 'ELSE' expected |
| 17 | - | 'ENDIF' expected |
| 18 | - | 'THEN' without corresponding 'IF' |
| 19 | - | 'ELSE' without corresponding 'IF' |
| 20 | - | 'ENDIF' without corresponding 'IF' |
| 21 | - | assignment operator '=' expected |
| 22 | - | wrong format for integer number |
| 23 | - | wrong format for real number |
| 24 | - | formula too complicated |
| 25 | - | error in arithmetic expression |

- 26 - internal compiler error
- 27 - identifier not valid
- 28 - unknown type identifier
- 29 - wrong input sign
- 30 - stack overflow of parser
- 31 - syntax error
- 32 - available memory exceeded
- 33 - index or index set not allowed
- 34 - error during dynamic storage allocation
- 35 - wrong number of indices
- 36 - wrong number of arguments
- 43 - number of variables different from declaration
- 44 - number of functions different from declaration
- 45 - END - sign not allowed
- 46 - Fortran code exceeds line
- 47 - **: domain error
- 48 - bad input format
- 49 - length of working array IWA too small
- 50 - length of working array WA too small
- 51 - ATANH: domain error
- 52 - LOG: domain error
- 53 - SQRT: domain error
- 54 - ASIN: domain error
- 55 - ACOS: domain error
- 56 - ACOSH: domain error
- 57 - LABEL defined more than once
- 58 - LABEL not found
- 59 - wrong index expression
- 60 - wrong call of the subroutine SYMINP
- 61 - wrong call of the subroutine SYMPRP
- 62 - compilation of the source file in GRAD-mode
- 63 - interpolation values not in right order
- 64 - not enough space for interpolation functions in subroutine REVCDE
- 65 - length of working array IWA in subroutine SYMFOR too small
- 66 - not enough interpolation values

Chapter 6

Model Functions and Equations

Model functions of the test examples are defined in the PCOMP language. The meaning of variables and functions is fixed by their serial order. Identifiers can be chosen arbitrarily.

6.1 Explicit Model Functions

To define model variables and explicit fitting functions in the PCOMP syntax, one has to follow certain guidelines for the declaration of parameters and functions, since the order in which these items are defined, is essential for the interface between the input file and the data fitting code. For defining variables, we need the following rules:

1. The first variable names are identifiers for the n independent parameters to be estimated, i.e., for p_1, \dots, p_n .
2. If a so-called concentration variable c exists, then a corresponding variable name must be added next.
3. The last variable name identifies the independent time variable t for which measurements are available.
4. Any other variables are not allowed to be declared.

Similarly, we have rules for the sequence by which model functions are defined:

1. First, r fitting criteria $h_1(p, t, c), \dots, h_r(p, t, c)$ must be defined depending on p, t , and optionally on c .
2. The subsequent m_r functions are the constraints $g_1(p), \dots, g_{m_r}(p)$, if they exist at all. They may depend only on the parameter vector p to be estimated.
3. Any other functions are not allowed to be declared.

The constants n , r , and m_r are defined in the database of **EASY-FIT**^{*ModelDesign*}. In addition to variables and functions, a user may insert further real or integer constants in the function input file according to the syntax rules of PCOMP.

Example: To illustrate the usage of symbolic function input, we consider an example. We have two explicit model functions

$$\begin{aligned} h_1(p, t) &= D \exp(-k_1 t) \ , \\ h_2(p, t) &= \frac{k_1 D}{k_1 - k_2} (\exp(-k_2 t) - \exp(-k_1 t)) \ . \end{aligned}$$

The corresponding input file is the following one:

```

C-----
C
C   Problem:  LKIN_X
C
C-----
C
*   VARIABLE
    k1, k2, D, t
C
*   FUNCTION h1
    h1 = D*EXP(-k1*t)
C
*   FUNCTION h2
    h2 = k1*D/(k1 - k2)*(EXP(-k2*t) - EXP(-k1*t))
C
*   END
C

```


6.2 Laplace Transformations

The input of variables and Laplace functions is very similar to the input of explicit model functions. Variables are:

1. The first variable names are identifiers for the n independent parameters to be estimated, p_1, \dots, p_n .
2. If a concentration variable exists, then a variable name must be added next that represents the concentration variable c .
3. The last variable name identifies the independent variable s in the Laplace space that corresponds to the time variable t after back-transformation, for which measurements are available.
4. Any other variables are not allowed to be declared.

Since constraints are not allowed, the only functions that can be declared, are r fitting criteria formulated as functions in the Laplace space, any functions $H_k(p, s, c)$ for $k = 1, \dots, r$, depending on p , s and c . Any other functions are not permitted. These functions are then transformed back to the original variable space in the time variable t .

The constants n and r are defined in the database of **EASY-FIT**^{ModelDesign} and must coincide with the corresponding numbers of variables and functions, respectively.

Example: To illustrate the usage of function input in the Laplace space, we consider

$$Y_1(s) = \frac{D}{s + k_1} \quad ,$$

$$Y_2(s) = \frac{k_1 D}{(s + k_1)(s + k_2)} \quad .$$

The functions are the Laplace transforms of two simple linear differential equations. If measurements are given for both functions, we define a model function file in the following way:

```
C-----
C
C   Problem:  LKIN_L
C
C-----
C
*   VARIABLE
    k1, k2, D, s
C
*   FUNCTION Y1
    Y1 = D/(s + k1)
C
*   FUNCTION Y2
    Y2 = k1*D/((s + k1)*(s + k2))
C
*   END
C
```

6.3 Systems of Steady State Equations

In this case, our system variables must be declared in the following order:

1. The first n names identify the n independent parameters to be estimated, p_1, \dots, p_n .
2. The subsequent m identifiers define state variables of the system of nonlinear equations, z_1, \dots, z_m .
3. If a so-called concentration variable c exists, a corresponding variable name must be added next.
4. The last name identifies the independent time variable t , for which measurements are available.
5. Any other variables are not allowed to be declared.

Model functions defining the algebraic equations, constraints, and fitting criteria are defined as follows:

1. The first m functions are the right-hand sides of the steady state equations, $s_1(p, z, t, c), \dots, s_m(p, z, t, c)$.
2. The subsequent m functions define starting values for solving the system of equations, which may depend on the parameters to be estimated, on the time variable, and eventually also on the concentration variable, $z_1^0(p, t, c), \dots, z_m^0(p, t, c)$.
3. Next, r fitting functions $h_1(p, z, t, c), \dots, h_r(p, z, t, c)$ must be defined depending on p, z, t , and c , where z denotes the state variables.
4. The final m_r functions are the constraints $g_j(p)$ for $j = 1, \dots, m_r$, if they are present in the model, depending on the parameter vector p to be estimated.
5. Any other functions are not allowed to be declared.

The constants n, r, m , and m_r are defined in the database of **EASY-FIT**^{*ModelDesign*}. In addition to variables and functions, a user may insert further real or integer constants in the function input file according to the guidelines of the language.

Example: We consider a simple example that is related to a receptor-ligand binding study with one receptor and two ligands. The system of equations is given in the form

$$\begin{aligned} z_1(1 + p_1 z_2 + p_2 z_3) - p_3 &= 0, \\ z_2(1 + p_1 z_1) - p_4 &= 0, \\ z_3(1 + p_2 z_1) - t &= 0. \end{aligned}$$

State variables are z_1 , z_2 , and z_3 . The parameters to be estimated, are p_1 , p_2 , p_3 and p_4 , i.e., $m = 3$ and $n = 4$. t is the independent model or time variable to be replaced by experimental data. The fitting criterion is $h(p, z, t) = p_4 - z_2$ and we use the starting values $z_1^0 = p_3$, $z_2^0 = p_4$ and $z_3^0 = t$ for solving the system of nonlinear equations.

```

C-----
C
C   Problem:  DYN_EQ
C
C-----
C
*   VARIABLE
    p1, p2, p3, p4, z1, z2, z3, t
C
*   FUNCTION g1
    g1 = z1*(1 + p1*z2 + p2*z3) - p3
C
*   FUNCTION g2
    g2 = z2*(1 + p1*z1) - p4
C
*   FUNCTION g3
    g3 = z3*(1 + p2*z1) - t
C
*   FUNCTION z1_0
    z1_0 = p3
C
*   FUNCTION z2_0
    z2_0 = p4
C
*   FUNCTION z3_0
    z3_0 = t
C
*   FUNCTION h
    h = p4 - z2
C
*   END
C

```

6.4 Ordinary Differential Equations

For defining variables, we need the following rules:

1. The first variables are identifiers for the n independent parameters to be estimated, p_1, \dots, p_n .
2. The subsequent s names identify the state variables of the system of ordinary differential equations, y_1, \dots, y_m .
3. If a concentration variable exists, then an identifier name must be added next that represents c .
4. The last variable name identifies the independent time variable t , for which measurements are available.
5. Any other variables are not allowed to be declared.

Similarly, we have rules for the sequence by which model functions are to be defined:

1. The first m functions are the right-hand sides of the system of differential equations, the functions $F_1(p, y, t, c), \dots, F_m(p, y, t, c)$.
2. The subsequent m functions define the initial values, which may depend on the parameters to be estimated, and the concentration variable, $y_1^0(p, c), \dots, y_m^0(p, c)$.
3. Next, r fitting functions $h_1(p, y, t, c), \dots, h_r(p, y, t, c)$ are defined depending on p, y, t , and c , where y denotes the state variable of the system of differential equations.
4. The final m_r functions are the constraints $g_j(p)$ for $j = 1, \dots, m_r$, if they exist at all, depending on the parameter vector p to be estimated.
5. Any other functions are not allowed to be declared.

The constants n , m , r , and m_r are defined in the database of **EASY-FIT**^{ModelDesign}. The last n_b of the n parameters to be estimated, are considered as switching points, if they have been declared to describe certain model changes. Also n_b , the number of constant or variable break points, must be defined a priori. In addition to variables and functions, a user may insert further real or integer constants in the function input file according to the guidelines of the language PCOMP.

Example: The example was introduced in Section 2.5 of Chapter 2. Although an explicit solution is easily obtained, we show here a possible implementation to illustrate the input of differential equations. The system is given by two equations of the form

$$\begin{aligned} \dot{y}_1 &= -k_1 y_1 & , \quad y_1(0) &= D \quad , \\ \dot{y}_2 &= k_1 y_1 - k_2 y_2 & , \quad y_2(0) &= 0 \quad . \end{aligned}$$

We assume that experimental data are available for both state functions $y_1(t)$ and $y_2(t)$, and define the corresponding PCOMP code as follows:

```

C-----
C
C   Problem: LKIN
C
C-----
C
*   VARIABLE
    k1, k2, D, y1, y2, t
C
*   FUNCTION y1_t
    y1_t = k1*y1
C
*   FUNCTION y2_t
    y2_t = k1*y1 - k2*y2
C
*   FUNCTION y1_0
    y1_0 = D
C
*   FUNCTION y2_0
    y2_0 = 0
C
*   FUNCTION h1
    h1 = y1
C
*   FUNCTION h2
    h2 = y2
C
*   END
C

```

6.5 Differential Algebraic Equations

The following order of PCOMP variables is required:

1. The first variable names are identifiers for n parameters to be estimated, p_1, \dots, p_n .
2. The subsequent m_d names identify the differential variables y_1, \dots, y_{m_d} .
3. The subsequent m_a names identify the algebraic variables z_1, \dots, z_{m_a} .
4. If a concentration variable exists, another identifier must be added next to represent c .
5. The last variable name defines the independent time variable t for which measurements are available.
6. Any other variables are not allowed to be declared.

Similarly, we have rules for the sequence by which the model functions are defined:

1. The first m_d functions define the differential equations, $F_1(p, y, z, t, c), \dots, F_{m_d}(p, y, z, t, c)$.
2. The subsequent m_a functions are the right-hand sides of the algebraic equations, i.e., functions $G_1(p, y, z, t, c), \dots, G_{m_a}(p, y, z, t, c)$.
3. Subsequently, m_d functions define initial values for the differential equations, which may depend on the parameters to be estimated, and the concentration variable, $y_1^0(p, c), \dots, y_{m_d}^0(p, c)$.
4. Then m_a functions define initial values for the algebraic equations, which may depend on the parameters to be estimated, and the concentration variable, $z_1^0(p, c), \dots, z_{m_a}^0(p, c)$.
5. Next r fitting functions $h_1(p, y, z, t, c), \dots, h_r(p, y, z, t, c)$ must be defined depending on p, y, z, t , and c , where y and z are the differential and algebraic state variables of the DAE.
6. The final m_r functions are the constraints $g_j(p)$, $j = 1, \dots, m_r$, if they exist. They may depend on the parameter vector p to be estimated.
7. Any other functions are not allowed to be declared.

The constants n , m_d , m_a , r , and m_r are defined in the database of **EASY-FIT**^{ModelDesign} and must coincide with the corresponding numbers of variables and functions, respectively. The last n_b fitting variables are considered as switching points, if they have been declared a priori to describe certain model changes. In addition to variables and functions, a user may insert further real or integer constants in the function input file according to the guidelines of the language PCOMP.

Example: We consider a modification of van der Pol's equation given in the form

$$\dot{y} = z \quad , \quad \dot{z} = a(1 - y^2)z \quad .$$

We choose the consistent initial values

$$y^0 = b \quad , \quad z^0 = b/(a(1 - b^2))$$

and consider a and b as parameters to be estimated. The fitting criteria are the solutions $y(t)$ and $z(t)$. The model input file has the following structure:

```

C-----
C
C      Problem: VDPOL
C
C-----
C
C      VARIABLE
C      a, b, y, z, t
C
C      FUNCTION y_t
C      y_t = z
C
C      FUNCTION alg_equ
C      alg_equ = y - a*(1 - y**2)*z
C
C      FUNCTION y0
C      y0 = b
C
C      FUNCTION z0
C      z0 = b/(a*(1 - b*b))
C
C      FUNCTION h1
C      h1 = y
C
C      FUNCTION h2
C      h2 = z
C
C      END
C

```

6.6 Time-Dependent Partial Differential Equations

For defining variables, we need the following rules:

1. The first variable names identify the n independent parameters to be estimated, p_1, \dots, p_n .
2. The subsequent names specify the state variables of the partial differential equations, u_1, \dots, u_{n_p} .
3. In a similar way, the names of the corresponding variables denoting the first and second spatial derivatives are to be declared in this order, u_{1x}, \dots, u_{n_px} and $u_{1xx}, \dots, u_{n_pxx}$.
4. Next, the names of n_c state variables belonging to coupled ordinary differential equations must be defined, v_1, \dots, v_{n_c} .
5. If flux functions are to be inserted into the right-hand side formulation of the PDE, then n_p identifiers for the flux and their spatial derivatives are to be given, f_1, \dots, f_{n_p} and f_{1x}, \dots, f_{n_px} .
6. Then a name is to be defined for the space or spatial variable x .
7. The last name identifies the independent time variable t for which measurements are available.
8. Any other variables are not allowed to be declared.

In a similar way, we have rules for the sequence by which the model functions are defined:

1. If flux functions are to be used, then $n_a n_p$ functions $f_1^i(p, u, u_x, x, t), \dots, f_{n_p}^i(p, u, u_x, x, t)$ defining the flux must be inserted, one set for each integration area, $i = 1, \dots, m_a$. They may depend on x, t, u, u_x , and p . When evaluating the right-hand side of model equations subsequently, then the values of these flux functions and their derivatives are passed to the identifier names and corresponding derivative variables declared in the variable section of the input file as outlined above.
2. Model functions defining the right-hand side of the partial differential equations

$$F_1^i(p, f^i, f_x^i, u, u_x, u_{xx}, v, x, t), \dots, F_{n_p}^i(p, f^i, f_x^i, u, u_x, u_{xx}, v, x, t)$$

are defined next, one set for each integration area, $i = 1, \dots, m_a$. Each function may depend on $x, t, v, u, u_x, u_{xx}, p$, and, optionally, also on the flux functions and their derivatives. In this case, the corresponding identifiers for fluxes and their derivatives as specified in the variable section, must be used in the right-hand side.

3. The corresponding initial values at time 0 are set next, $u_0^i(p, x)$, $i = 1, \dots, m_a$. They depend on x and p , and are given for each integration area separately.
4. Next, the n_c coupled differential equations must be defined in the order given by the series of coupling points, i.e., functions $G_j(p, u, u_x, u_{xx}, v, t)$, $j = 1, \dots, n_c$, where the state variable u is evaluated at a given discretization line together with its first and second spatial derivatives.
5. Then initial values of the coupled ordinary equations at time 0 are defined, $v_0^j(p)$, $j = 1, \dots, n_c$.
6. Subsequently, n_b Dirichlet transition and boundary conditions are set in the order given by the area data, first left, then right boundary functions $c_1(p, u, v, t), \dots, c_{n_b}(p, u, v, t)$, where function values of u at the left or right end point of an integration area are inserted.
7. Neumann transition and boundary conditions for spatial derivatives follow in the order given by the area data, $\hat{c}_1(p, u, u_x, v, t), \dots, \hat{c}_{\hat{n}_b}(p, u, u_x, v, t)$. Again, the function values of u or u_x at a suitable end point of an integration area are inserted.
8. Moreover, r fitting criteria must be defined, any functions $h_1(p, u, u_x, u_{xx}, v, t), \dots, h_r(p, u, u_x, u_{xx}, v, t)$. u is defined at the corresponding spatial fitting point.
9. The final m_r functions are the constraints $g_1(p), \dots, g_{m_r}(p)$, if they exist. They may depend on the parameter vector p to be estimated.
10. Any other functions are not allowed to be declared.

All constants and in particular the structure of the integration interval are defined in the database of **EASY-FIT**^{ModelDesign}, and must coincide with the corresponding numbers of variables and functions, respectively. In addition, a user may insert further real or integer constants in the function input file according to the guidelines of PCOMP.

Example: We consider a simple example, where Fourier's first law for heat conduction leads to the equation

$$u_t = u_{xx}$$

defined for $0 < t \leq 0.5$ and $0 < x < 1$. Boundary conditions are

$$u(0, t) = u(1, t) = 0$$

for $0 \leq t \leq 0.5$ and the initial values are

$$u(x, 0) = \sin\left(\frac{\pi x}{L}\right)$$

for $0 < x < 1$ and $0 \leq L \leq 1$. In addition, we are interested in the total amount of heat at the surface $x = 0$ given by the equation

$$\dot{v} = -k \cdot \frac{\pi}{L} \cdot e^{-\frac{\pi^2}{L^2} \cdot t}$$

with initial heat

$$v_0 = \frac{k \cdot L}{\pi} \quad .$$

Function v serves also as our fitting criterion. Parameters to be estimated, are L and k . The corresponding PCOMP input file is:

```
C-----
C
C   Problem: HEAT
C
C-----
C
C   REAL CONSTANT
C   pi = 3.1415926535
C
C   VARIABLE
C   L, k, u, u_x, u_xx, v, x, t
C
C   FUNCTION u_t
C   u_t = u_xx
C
C   FUNCTION u0
C   u0 = DSIN(pi*x/L)
C
C   FUNCTION v_t
C   v_t = -k*pi/L*DEXP(-(pi/L)**2*t)
C
C   FUNCTION v0
C   v0 = k*L/pi
C
C   FUNCTION u_left
C   u_left = 0
C
C   FUNCTION u_right
C   u_right = 0
C
C   FUNCTION h
C   h = v
C
C   END
C
```

6.7 Partial Differential Algebraic Equations

Very similar to the definition of data fitting problems based on partial differential equations outlined in the previous section, we have to define fitting criteria, differential equations, initial and boundary conditions, coupling and transition equations and constraints in a suitable format. For defining variables, we need the following rules:

1. The first names are the identifiers for n independent parameters to be estimated, p_1, \dots, p_n .
2. The subsequent names identify the n_p state variables of the system, u_1, \dots, u_{n_p} , where first the differential, then the algebraic variables must be listed.
3. In a similar way, the corresponding variables denoting the first and second spatial derivatives of differential and algebraic variables are to be declared in this order, u_{1x}, \dots, u_{n_px} and $u_{1xx}, \dots, u_{n_pxx}$.
4. Next, names of n_c variables belonging to coupled differential algebraic equations are defined, v_1, \dots, v_{n_c} , where first the differential, then the algebraic variables must be given.
5. If flux functions are to be inserted into the right-hand side formulation of the PDAE, then n_p identifiers for the fluxes and their spatial derivatives are given, f_1, \dots, f_{n_p} and f_{1x}, \dots, f_{n_px} .
6. Then a name is to be defined for the space or spatial variable x .
7. The last name identifies the independent time variable t for which measurements are available.
8. Any other variables are not allowed to be declared.

Model functions are defined in the following format:

1. If flux functions are to be used, then $n_a n_p$ functions $f_1^i(p, u, u_x, x, t), \dots, f_{n_p}^i(p, u, u_x, x, t)$ defining the flux are inserted, one set for each integration area, $i = 1, \dots, m_a$. They may depend on x, t, u, u_x , and p .
2. Functions for the right-hand side of partial differential equations

$$F_1^i(p, f^i, f_x^i, u, u_x, v, x, t), \dots, F_{n_p}^i(p, f^i, f_x^i, u, u_x, v, x, t)$$

are defined next, one set for each integration area, $i = 1, \dots, m_a$. Each function may depend on $x, t, v, u, u_x, u_{xx}, p$, and, optionally, also on the flux functions and their derivatives. First, the differential equations, then the algebraic equations must be defined.

3. Then corresponding initial values at time 0 must be set, $u_0^i(p, x)$, $i = 1, \dots, m_a$, where first initial values for the differential and then for the algebraic equations must be declared. They depend on x and p , and are given for each integration area separately.
4. Next, n_c coupled differential equations followed by the coupled algebraic equations are specified in the order given by the series of coupling points, i.e., the functions $G_j(p, u, u_x, u_{xx}, v, t)$, $j = 1, \dots, n_c$, where the state variable u is evaluated at a given discretization line together with its first and second spatial derivatives.
5. The corresponding initial values of the coupled ordinary differential algebraic equations at time 0 must be defined, $v_0^j(p)$, $j = 1, \dots, n_c$, in the same order.
6. Then n_b Dirichlet transition and boundary conditions must be set in the order given by the area data, first left, then right boundary, $c_1(p, u, v, t)$, \dots , $c_{n_b}(p, u, v, t)$, where function values of u at the left or right end point of an integration area are inserted.
7. Subsequently, transition and boundary conditions for spatial derivatives must be defined in the order given by the area data, i.e., the functions $\hat{c}_1(p, u, u_x, v, t)$, \dots , $\hat{c}_{\hat{n}_b}(p, u, u_x, v, t)$. Again, the function values of u or u_x at a suitable end point of an integration area are inserted.
8. Moreover, r fitting criteria have to be given, any functions $h_1(p, u, u_x, u_{xx}, v, t)$, \dots , $h_r(p, u, u_x, u_{xx}, v, t)$. u is defined at the corresponding spatial fitting point.
9. The final m_r functions are the constraints $g_1(p)$, \dots , $g_{m_r}(p)$, if they exist. They may depend on the parameter vector p to be estimated.
10. Any other functions are not allowed to be declared.

All constants and in particular the structure of the integration interval are defined in the database of **EASY-FIT**^{ModelDesign} and must coincide with the corresponding numbers of variables and functions, respectively. Note that initial values for algebraic variables serve only as starting values for applying a nonlinear programming algorithm to compute consistent initial values of the discretized DAE system.

Example: We consider a very simple fourth-order partial differential equation obtained from successive differentiation of $u(x, t) = ae^{-\pi^4 t} \sin(\pi x)$,

$$u_t = -au_{xxxx}$$

or, equivalently, two second-order differential algebraic equations

$$\begin{aligned} u_t &= -av_{xx} \ , \\ 0 &= v - u_{xx} \end{aligned}$$

defined for $0 \leq x \leq 1$ and $t \geq 0$. Initial values are $u(x,0) = \sin(\pi x)$ and $v(x,0) = -\pi^2 \sin(\pi x)$ and boundary values are $u(0,t) = u(1,t) = v(0,t) = v(1,t) = 0$ for all $t \geq 0$. Function u is a possible fitting criterion and a an unknown parameter to be estimated from experimental data. The corresponding PCOMP input file is:

```
C-----
C
C   Problem: PDEA4
C
C-----
C
*   REAL CONSTANT
    pi = 3.1415926535
C
*   VARIABLE
    a, u, v, u_x, v_x, u_xx, v_xx, x, t
C
*   FUNCTION u_t
    u_t = -a*v_xx
C
*   FUNCTION alg_equ
    alg_equ = v - u_xx
C
*   FUNCTION u_0
    u_0 = sin(pi*x)
C
*   FUNCTION v_0
    v_0 = -pi**2*sin(pi*x)
C
*   FUNCTION u_left
    u_left = 0
C
*   FUNCTION u_right
    u_right = 0
C
*   FUNCTION v_left
    v_left = 0
C
*   FUNCTION v_right
    v_right = 0
C
*   FUNCTION h
    h = u
C
*   END
C
```

Chapter 7

Problem Data and Solution Tolerances

To start a parameter estimation or simulation run, **EASY-FIT**^{*ModelDesign*} generates an input file for MODFIT or PDEFIT, respectively. The file has to contain all data that are passed to the numerical algorithm, i.e., measurement values, starting values for the parameters to be estimated, solution tolerances etc.

We have to distinguish between general problem data that are independent from the type of the mathematical model, and the model dependent information needed to start a parameter estimation run. Data are collected within the main form of **EASY-FIT**^{*ModelDesign*} and are kept in the data base until they are deleted or changed.

7.1 Model Independent Information

First, a user has to provide database information that is independent from the underlying model. Part of the data required is optional and only needed to document the input, part is mandatory to set tolerances and options depending on the choice of an optimization routine.

7.1.1 Problem Name

The string to be inserted here, is the unique key number to identify the parameter estimation problems in the database. Thus, the string must be non-empty and unique. It is not possible to change the name subsequently. If one needs to alter the problem name, one should copy the actual problem to another one with the desired name, and delete the old one. The problem name serves to define file names in the form <name>.FUN for model functions in the problem directory of the actual **EASY-FIT**^{*ModelDesign*} version, among many others. Therefore, the name must satisfy the usual DOS conventions, i.e., must be an alpha-numeric string with up to 8 characters.

EASY-FIT ModelDesign

Actual Problem: **SOIL** Prof. Klaus Schittkowski Bayreuth, Germany Version: 5.00 (Jan 2009)

Model Data | Experimental Data

Model: **PDE**
 Information: Diffusion of water through soil, convection and dispersion
 Project Number: Demo Unit for X-Values: t
 User Name: Schittkowski Unit for Y-Values: Names
 Measurement Set: Experimental Unit for Z-Values: x
 Date: 14.12.1998
 Memo: van Genuchten M.T., Wierenga P.J. (1976): Mass transfer studies in sorbing porous media. 1. Analytical solutions, Soil Sci. Soc. Am. Journal, Vol. 44, 892-898
 Anderson F., Olsson B. (eds.) (1985): Lake Gaadsj'on. An acid forest lake and its catchment, Ecological Bulletins, Vol. 37

no	name	lower bound	starting value	upper bound	optimal	PL
1	pm	1.00E-03	1.0000E+00	1.00E+04	1.1438E+01	3
2	pim	1.00E-05	1.0000E+00	1.00E+05	6.0377E+00	2
3	Dm	1.00E-05	1.0000E+02	1.00E+05	3.7099E+02	1

Parameters to be Estimated: Update Simulation Data Fitting Experimental Design

Model Type:

- explicit - explicit model functions with concentration values
- system - system of steady state equations
- Laplace - Laplace formulation of model functions with concentration values and internal backtransformation
- ODE - system of ordinary differential equations with initial values, concentration values, and switching points
- DAE - system of differential algebraic equations up to index 3 with initial values, concentration values, and switching points
- PDE - system of one-dimensional, time-dependent partial differential equations with initial and boundary values, disjoint integration areas, coupled ODE's, flux functions and switching points
- PDAE - same as for PDE, but with additional algebraic partial differential equations and coupled DAE's

Model Parameters: Language: PCOMP FORTRAN
 Norm: Sum of Absolute Residual Values
 Sum of Squared Residuals
 Maximum of Residual Values

Record: 1173 of 1398
 Model name (optional)

Figure 7.1: Main Form

7.1.2 Documentation Text

There are a few information strings that are useful to identify different parameter estimation problems particularly if a sequence of experimental data sets is to be processed. These strings are inserted in the reports generated by **EASY-FIT^{ModelDesign}** and the plot output.

Note that the proposed meaning of the items has no additional impact on the problem within **EASY-FIT^{ModelDesign}**. A particular advantage is the possibility to define a filter on the actual database and to select certain subsets of problems with predetermined properties.

Model name: Arbitrary string to identify the type of a model.

Information: Long information string, up to 80 characters.

Project number: Could contain project or any related information.

User name: Is set to the user name in the database when generating a new problem.

Measurement set: Identification of measurement data.

Date: Inserted automatically when generating a new problem.

Unit for X-values: Any string for identifying the X- or time axis in plots.

Unit for Y-values: Any string for identifying the Y- or measurement axis in plots. In a separate window also names for individual measurement sets are defined.

Unit for Z-values: Any string for identifying the Z- or concentration axis in 3D-plots.

Memo: Text field for input of additional documentation text.

Moreover, fitting criteria can be identified by some strings used in the generated reports and plots.

7.1.3 Parameters to be Estimated

The names for the optimization parameters to be estimated, their upper and lower bounds and their starting values must be defined in form of table records, where the initial value must not be smaller than the lower bound or greater than an upper bound. The variable name is arbitrary and used only for generating readable output.

Note that the order in which the variables are defined, must coincide with the order in which they are used in the model function part, i.e., either a Fortran code or a PCOMP input file. To guarantee the correct order in the underlying database, we need corresponding order information. Thus, the first column must contain the sequel number of the variables.

After a successful data fitting run, the optimal parameter values and, if a positive significance level has been set, also their significance levels are listed. Since all numerical executions are run as separate processes, their termination is checked by the user interface only at special occasions, e.g., in case of generating a report. To see these data immediately after execution of MODFIT.EXE or PDEFIT.EXE, one has to click on the command bottom 'update table'.

7.1.4 Input Type of Model Functions

Basically there are two possibility to define the nonlinear model functions:

PCOMP: All variables and functions are declared in a Fortran-similar syntax, where the order of the variables must coincide with the order used in the database. For function input one has to follow the guidelines outlined in Chapter 5. Particular advantage is the possibility to let derivatives be computed automatically during run time, i.e., without additional approximation or round-off errors.

Fortran: Alternatively, a user has the choice to prepare model functions in form of Fortran statements, if the PCOMP syntax is too restrictive or if he wants to accelerate the function evaluation. The order in which functions must be coded, and the organization of

corresponding subroutine arguments, is described within initial comments of the frame inserted automatically when generating a new problem. There is only one subroutine for the evaluation of all model functions and gradients depending on an input flag. If the subroutine is called with zero flag, then additional statements or subroutine calls can be inserted to prepare data before starting the optimization cycle. In case of explicit fitting functions and dynamical systems of equations, analytical gradient evaluation may be omitted. Then numerical approximations based on forward differences are applied. In case of ordinary differential equations, gradients must be programmed only if an ODE solver is to be executed that requires the computation of derivatives either by an implicit method or by internal evaluation of derivatives.

7.1.5 Numerical Analysis

EASY-FIT^{*ModelDesign*} allows to perform a simulation with respect to given set of parameter values, to start an iterative data fitting run, or to compute an optimal experimental design.

Simulation: A numerical analysis either at the set of given parameter values or the parameter values obtained from a previous data fitting run is performed. Moreover, plot data are generated.

If the significance level for determining confidence intervals is positive, also a statistical analysis is performed. Proceeding from the assumption that the model is sufficiently linear in a neighborhood of an optimal solution vector and that all experimental data are Gaussian and independent, the following statistical data are evaluated:

- Variance/covariance matrix of the problem data
- Correlation matrix of the problem data
- Estimated variance of residuals
- Confidence intervals for the individual parameters subject to the significance levels 1%, 5% or 10%, respectively.

The number of model parameters for which the covariance matrix and the confidence intervals are computed, is typically the number of all given parameters. But there are situations where these parameters are divided into model and design parameters, for example, and where the statistical data are to be evaluated only for a certain subset. Thus, the number of these variables can be restricted to those, for which a statistical analysis is computed. The given value corresponds to the first ones found in the table, and are called the model variables.

Data Fitting: The mathematical background of the applied algorithms is described in Schittkowski [429]. The basic idea is to introduce additional variables and equality constraints, and to solve the resulting constrained nonlinear programming problem by the sequential quadratic programming algorithm NLPQLP of Schittkowski [427, 440]. It can be shown that

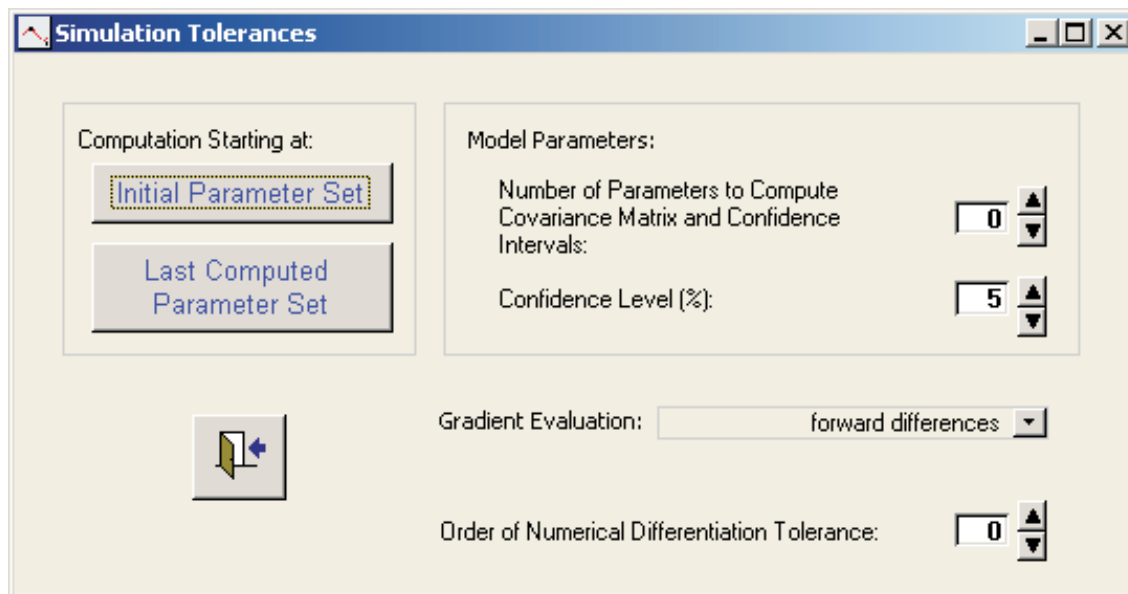


Figure 7.2: Simulation Tolerances

typical features of a special purpose method are retained, the combination of a Gauss-Newton and a quasi-Newton search direction in case of a least squares problem, see Schittkowski [429]. The additional variables and equality constraints are substituted in the quadratic programming subproblem, so that calculation time is not increased significantly by this approach. In case of minimizing a sum of absolute function values or the maximum of absolute function values, the problem is transformed into a smooth nonlinear programming problem. NLPLSQ [446] is capable to handle any additional linear or nonlinear equality or inequality constraints.

Since the number of variables increases with the number of experimental data, least squares problems are treated as general nonlinear optimization problems, if the number of data points exceeds 500. The total number of iterations might increase, but the calculation time per iteration is decreased.

Min-max problems are transformed into a general smooth optimization problem with only one additional variable and solved by the code NLPINF [448]. A sophisticated active set strategy is applied in case of more than 500 experimental data, i.e., more than 1,000 nonlinear constraints. The modified SQP code is called NLPQLB, see Schittkowski [442].

Experimental Design: Experimental design helps to find suitable values for additional so-called design parameters, for example initial concentrations or input feeds, which have to be set before conducting experiments. Various experimental design methods have been discussed in the literature before, see e.g. Winer, Brown and Michels [551], Ryan [414], Rudolph and Herrendörfer [412], Baltes et al. [19], or Lohmann et al. [296]. Since the confidence in-

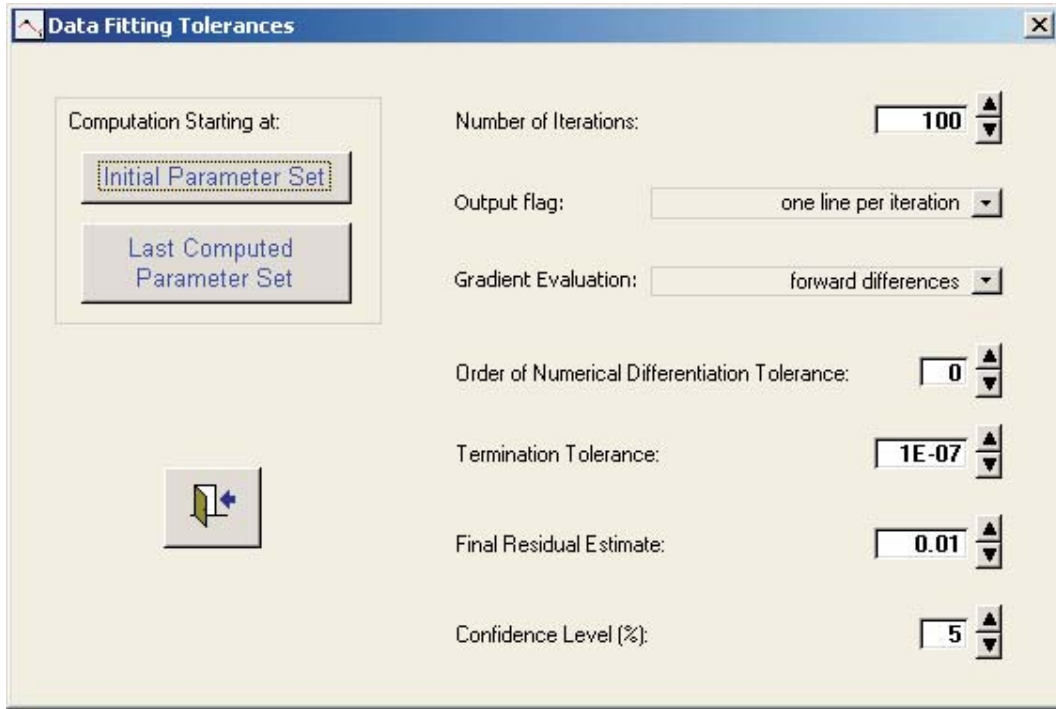


Figure 7.3: Data Fitting Tolerances

tervals mentioned above are mainly determined by the diagonal elements of the covariance matrix, a possible objective function is the trace of this matrix, see Schittkowski [441] for more details and a couple of case studies.

A further option is to locate experimental time values. Especially in case of time expensive experiments, it is highly desirable to minimize their number and to conduct experiments only within relevant time intervals. Thus, we add artificial weight factors to the observations at a predefined, relatively dense grid specified in advance. These weights are considered then as design parameters. A particular advantage is that derivatives subject to weights are obtained without additional computational efforts.

Special emphasis is given to efficient computation of derivatives, where first and second order partial derivatives of the model function of our dynamical system are approximated by forward differences. The nonlinear constrained optimization problems are solved by the SQP code NLPQLP, see Schittkowski [440].

7.1.6 Optimization Tolerances

A couple of tolerances and bounds are required to start the optimization algorithms for constrained nonlinear data fitting and experimental design. The database provides suitable default values that can be used, when a new problem is generated.

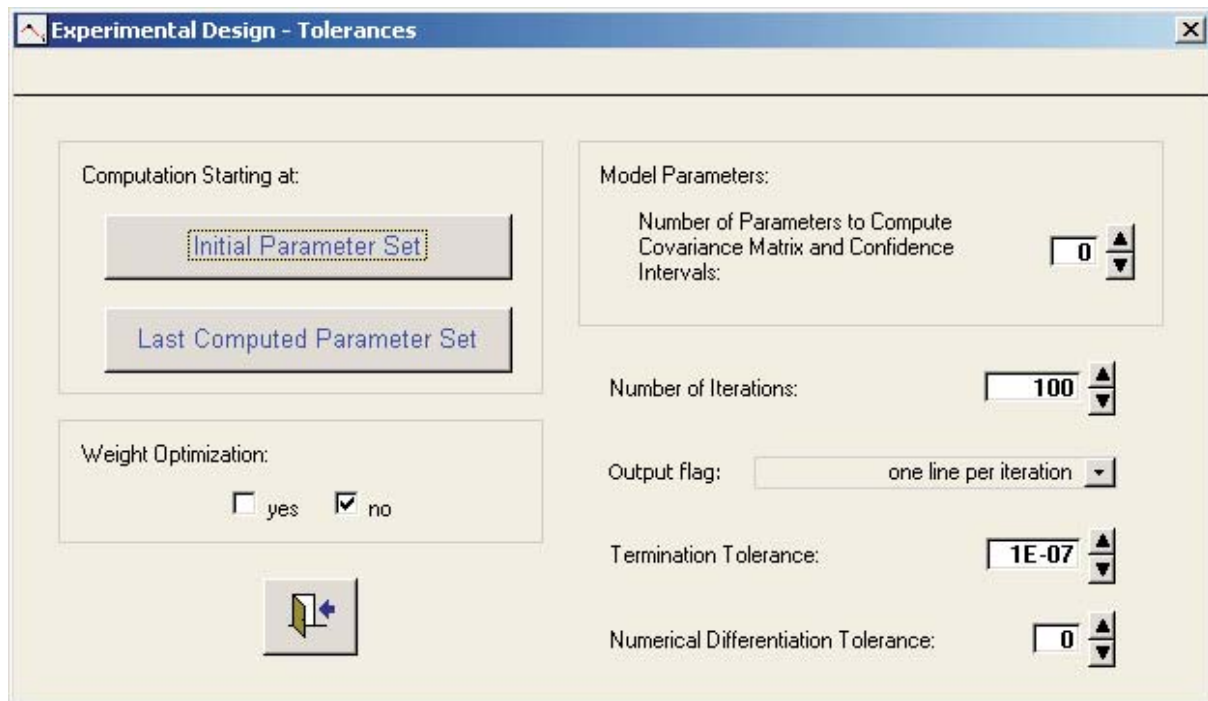


Figure 7.4: Experimental Design Tolerances

Number of Iterations: A reasonable upper limit for the number of iterations is required.

Data Fitting : One evaluation of the Jacobian matrix of the data fitting function with respect to the variables to be estimated.

Experimental Design : One evaluation of the covariance matrix of the experimental design performance measure.

It is recommended to start with a sufficiently low number, say 50. If the resulting answer is not as precise as wanted, a restart can be performed where the last computed iterates are inserted as starting values for another optimization run.

Output Flag: There is the possibility to control the desired output shown on the screen.

- 1 only the iteration number and the actual residual value displayed
- 0 no output
- 1 only final convergence analysis
- 2 one line per iteration
- 3 more detailed output information per iteration
- 4 in addition, also line search data are displayed

Output level 2 is recommended. The original output of the selected least squares algorithm is directed to a file with extension HIS depending on the print flag chosen. Only the residual values are displayed on screen in a DOS window executing the numerical code. However,

one has to be a bit familiar with the underlying mathematical theory to understand the data in detail. For the meaning of the parameters displayed, it is necessary to read the corresponding user guides.

Gradient Evaluation: **EASY-FIT**^{*ModelDesign*} comes with two options to define model functions: Fortran code to be compiled, or the PCOMP modeling language. In the latter case, gradients are evaluated automatically and are inserted whenever possible, for example for explicit, Laplace and steady-state models, calculation of consistent initial values, or for differentiation of right-hand side of an ODE/DAE subject to state variables. If a model is implemented in Fortran, it is recommended to provide all gradients analytically, i.e., to generate them by hand.

However, derivatives are not available analytically for the outer parameter estimation algorithms in case of differential equations. The only exception is the usage of internal numerical differentiation when executing the explicit solver. In all other cases, derivatives are approximated by numerical differences. The following derivative calculations are available:

- 0 - analytical differentiation
- 1 - forward differences
- 2 - two-sided differences
- 3 - five point approximation formula

It is important to know that forward differences require n , two-sided differences $2n$, and the five point formula $4n$ additional integrations, where n is the number of parameters to be estimated. If PCOMP is used, the value 0 should be inserted in case of explicit, Laplace, and steady-state systems, in all other situations a positive value.

Numerical Differentiation Error: The corresponding tolerance needed to compute these approximation, is defined separately in form of an integer that estimates the number of correct digits. To give an example, an entry of 5 leads to a perturbation of 10^{-5} for approximating the gradients. If this parameter is set to zero, a suitable tolerance is internally computed depending on the order of the differentiation formula and an estimated accuracy by which the data fitting function is evaluated.

Termination Tolerance: The tolerance is used to stop the algorithm as soon as some internal termination conditions are satisfied, and should not be smaller than the accuracy by which derivatives are computed (1.0E-6).

Final Residual Estimate: A rough estimate for the expected final residual value can be inserted in case of least squares norms (0.01). If the code breaks down after very few iterations, a larger step could lead to shorter steps and a more stabilized procedure.

Confidence Level: If the significance level for determining confidence intervals is positive, a statistical analysis is performed. Similar to a simulation run, the following statistical data are evaluated, where the termination tolerance is used for determining the rank of the correlation matrix:

- Variance/covariance matrix of the problem data

- Correlation matrix of the problem data
- Estimated variance of residuals
- Confidence intervals for the individual parameters subject to the significance levels 1%, 5% or 10%, respectively.

In addition, significance levels are evaluated to identify the significance of parameters and to help to decide upon questions like

- which parameters can be identified,
- which parameters should be kept fixed,
- whether additional experimental data must be required.

Overdetermined data fitting problems often lead to unstable and slow convergence of Gauss-Newton-type least squares algorithms. The idea is to successively eliminate parameters until (3.9) is satisfied. The cycle is terminated in one of the following situations:

1. The smallest eigenvalue of the Fisher information matrix is smaller than the given confidence level.
2. The parameter correlations are reduced by 25 %.
3. None of the above termination reasons are met and all parameters have been eliminated.

Level 1 corresponds to the first eliminated variables, level 2 to the second, etc. The final level can be assigned to several parameters indicating a group of identifiable parameters.

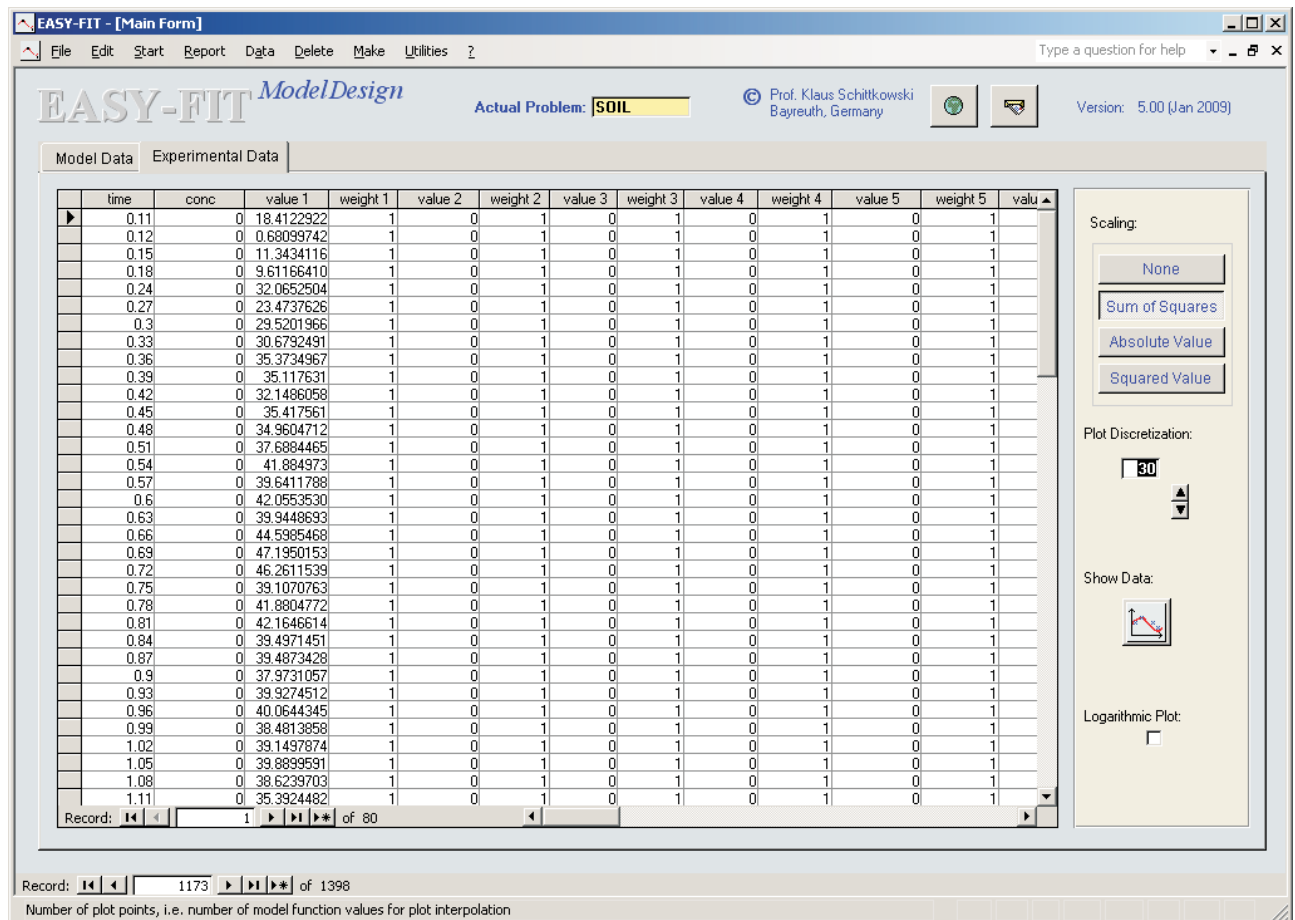


Figure 7.5: Measurement Data

7.1.7 Scaling

In addition to the individual weights that must be given by a user, it is possible to select a global scaling strategy:

- no additional scaling
- division of residuals by square root of sum of squares of all measurement values
- division of each single residual by corresponding absolute measurement value
- division of each single residual by corresponding squared measurement value

7.1.8 Number of Plot Points

Plots of model functions are generated from a number of function evaluations and a corresponding interpolation, linearly or by splines depending on the graphics system used. The number of points where model function values are to be evaluated by a final simulation run, is to be inserted. The higher the number, the more function evaluations are required, the more lines are displayed for 3D-plots in case of a PDE, and the more accurate is the curve.

Note, however, that in case of an ODE, DAE and PDE, only one integration is performed over the time axis, and the required solution values are retrieved from intermediate iterates of the algorithm by so-called dense output, i.e., by internal interpolation. Thus, the calculation time is not increased dramatically in case of a larger value.

The number of plot points must not be bigger than the parameter MAXPLT in the include file of the corresponding analysis code and must not exceed the maximum number of time values.

7.1.9 Logarithmic Plot

It is possible to require a logarithmic scale of the time axis in function and data plots. In this case, a plot is generated from the first to the last measurement value, not beginning at $t = 0$. Consequently, the corresponding time values must be positive.

7.1.10 Experimental Data

If l denotes the number of experiments, then one has to insert l records containing the following data:

Column headed by time: Input of measurement time t_i , $i = 1, \dots, l$. A set of experimental time values has to be repeated for each concentration value. If more than one measurement set is available, we assume that experimental data are available for all time values. Otherwise, experiments can be eliminated by zero weights. If the smallest time value is less than zero,

then the integration of an ordinary or partial differential equation is started at the point given. Otherwise, integration is always beginning at 0.

Alternatively, a fitting criterion might depend on time and concentration variables, where each time value corresponds to one concentration value, which may all be different. This option is not available for models based on partial differential equations.

Column headed by conc(entrations): Input of concentration values c_j . There must be one and the same concentration value for one series of time values. If more than one concentration value is available, we assume that all sets proceed from the same series of time values and the same concentration value. Otherwise, experiments must be deleted by zero weights.

If a fitting criterion depends on time and concentration variables, where each time value corresponds to one concentration value, then a suitable concentration value must be assigned to each time value. This option is not available for models based on partial differential equations.

Column headed by value: Input of measurement value y_{ij}^k for the actual measurement set. Any legal format for real numbers is allowed.

Column headed by weight: Input of nonnegative weight factor w_{ij}^k for actual measurement set. If the weight is set to zero, the corresponding measurement value is not taken into account.

7.1.11 Data Fitting Norm

Experimental data can be fitted in three different norms:

L_2 -norm: The classical least squares norm minimizes the sum of all squares of differences between model function and measurement values.

L_1 -norm: In this case, the sum of absolute values of all differences between model function and measurement values is minimized.

L_∞ -norm: The maximum of absolute values of all differences between model function and measurement values is minimized.

7.2 Model Dependent Information

For each of the parameter estimation models of **EASY-FIT**^{Model Design}, additional data are required depending on the model structure as outlined in Chapter 5.

7.2.1 Model Data for Explicit Functions

For the execution of the numerical analysis program MODFIT with explicitly given model functions, we need some integers that cannot be retrieved from the model function file or other data.

Number of Measurement Sets: The number of measurement sets must coincide with the number of data sets as given in the input table for experimental data. Note that a data set corresponds to two input columns in the table for values and weights.

Number of Concentration Values: The number of concentrations must coincide with the number of concentrations as given in the input table for experimental data. If the value inserted is positive and the PCOMP input language is used, then a concentration variable must be declared in the model function file. If -1 is inserted, it is supposed that the fitting criteria depend on an additional concentration variable, and that one concentration value is assigned to each time value.

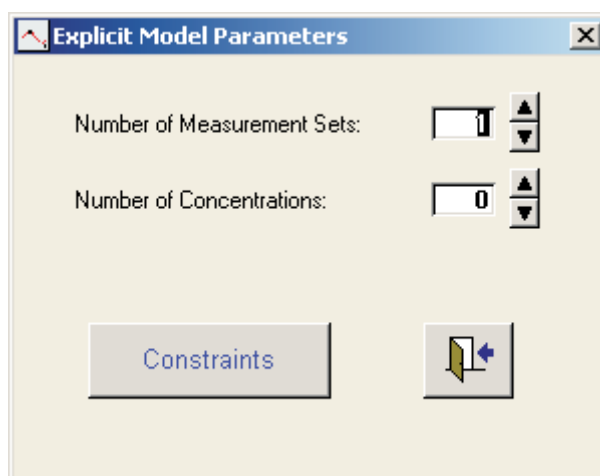


Figure 7.6: Parameters for Explicit Model Functions

Constraints: Restrictions are allowed for explicit model functions and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and some of the given experimental time and concentration values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first

in the input file for model functions. The table allows to define time and concentration values, for which constraints are to be defined:

Order:	Serial order number of constraints. Equality constraints must be defined first.
Name:	Arbitrary name for the constraint, to be printed in reports.
Time:	Corresponding time value at which a constraint is to be evaluated. Note that the time values are rounded to the nearest experimental time value, to avoid a complete re-integration of the system. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.
c/x-value:	Corresponding concentration value at which a constraint is to be evaluated. Note that concentration values are rounded to the nearest experimental concentration value. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental concentration value does not exist.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or concentration parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

Constraints

Number of Equality Constraints:

	order	name	time	c/x - value
▶	1	g	5	0
	2	g	6	0
	3	g	7	0
	4	g	8	0
	5	g	9	0
	6	g	10	0
	7	g	11	0
	8	g	12	0
	9	g	13	0
	10	g	14	0
	11	g	15	0
	12	g	16	0
	13	g	17	0
	14	g	18	0
	15	g	19	0
	16	g	20	0
	17	g	21	0
	18	g	22	0
	19	g	23	0
	20	g	24	0
	21	g	25	0
*	1	q		

Record: of 21

Figure 7.7: Constraints

7.2.2 Model Data for Steady State Equations

For the execution of the numerical analysis program MODFIT that estimates parameters in dynamical systems of equations, we need some integers that cannot be retrieved from the model function file or other data. Systems of nonlinear equations must be solved for each function and/or gradient evaluation required by the parameter estimation method, moreover, for each experimental time and concentration value separately. The system is treated as a general mathematical optimization problem and solved by the Fortran code NLPQLP, see Schittkowski [427, 440, 449]. The starting values of an optimization cycle must be predetermined by the user in the input file for model functions. They may depend on the parameters of the outer optimization problem.

Steady State Model Parameters

Number of Algebraic Functions: 2

Number of Measurement Sets: 1

Number of Concentrations: 0

Iteration Bound for Equation Solver: 50

Line Search Iteration Bound: 8

Output Flag for Equation Solver: 0

Final Accuracy for Equation Solver: 1E-10

Constraints

Figure 7.8: Parameters for Steady State Equations

Number of System Functions: The number of functions of the algebraic system must coincide with the number of functions to be declared in the model function file. The fitting criteria are not counted.

Number of Measurement Sets: The number of measurement sets must coincide with the number of data sets as given in the input table for experimental data. Note that a data set corresponds to two input columns in the table for values and weights.

Number of Concentration Values: The number of concentrations must coincide with the number of concentrations as given in the input table for experimental data. If the value

inserted is positive and the PCOMP input language is used, then a concentration variable must be declared in the model function file. If -1 is inserted, it is supposed that the fitting criteria depend on an additional concentration variable, and that one concentration value is assigned to each time value.

Maximum Number of Iterations for Solving Nonlinear Equations: For solving the internal systems of equations by NLPQLP, the maximum number of iterations is required. Usually a relatively small number of iterations is performed. However, numerical instabilities could require a larger bound (50).

Maximum Number of Line Search Iterations for Solving Nonlinear Equations: An internal line search is performed which requires additional function evaluations. One has to define a reasonably small bound (8).

Print Flag for Solving Nonlinear Equations: The output generated by the code NLPQLP, consists of an iteration summary obtained for 2, 3 or 4 or of an final optimization summary for 1. It is recommended to suppress all output by inserting the value 0.

Termination Accuracy for Solving Nonlinear Equations: It is recommended to use a relatively small termination tolerance for NLPQLP to get very precise function and gradient values for the outer optimization algorithm (1.0E-10). If, however, some warnings and errors are reported by MODFIT, one should try a larger value.

Constraints: Restrictions are allowed for explicit model functions and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and some of the given experimental time and concentration values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first in the input file for model functions. The table allows to define time and concentration values, for which constraints are to be defined:

- Order: Serial order number of constraints. Equality constraints must be defined first.
- Name: Arbitrary name for the constraint, to be printed in reports.
- Time: Corresponding time value at which a constraint is to be evaluated. Note that the time values are rounded to the nearest experimental time value, to avoid a complete re-integration of the system. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.
- c/x-value: Corresponding concentration value at which a constraint is to be evaluated. Note that concentration values are rounded to the nearest experimental concentration value. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental concentration value does not exist.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or concentration parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

Constraints				
Number of Equality Constraints: <input type="text" value="0"/>				
	order	name	time	c/x - value
▶	1	g	5	0
	2	g	6	0
	3	g	7	0
	4	g	8	0
	5	g	9	0
	6	g	10	0
	7	g	11	0
	8	g	12	0
	9	g	13	0
	10	g	14	0
	11	g	15	0
	12	g	16	0
	13	g	17	0
	14	g	18	0
	15	g	19	0
	16	g	20	0
	17	g	21	0
	18	g	22	0
	19	g	23	0
	20	g	24	0
	21	g	25	0
*	1	q		
Record: <input type="text" value="1"/> of 21				

Figure 7.9: Constraints

7.2.3 Model Data for Laplace Transformations

For the execution of the numerical analysis program MODFIT for models in the Laplace space, we need some further data that cannot be retrieved from the model function file or other information.

Number of Measurement Sets: The number of measurement sets must coincide with the number of data sets as given in the input table for experimental data.

Number of Concentration Values: The number of concentrations must coincide with the number of concentrations as given in the input table for experimental data. If the value inserted is positive and the PCOMP input language is used, then a concentration variable must be declared in the model function file. If -1 is inserted, it is supposed that the fitting criteria depend on an additional concentration variable, and that one concentration value is assigned to each time value.

Number of Iterations for Back-Transformations: MODFIT is capable solve parameter estimation problems, where the model functions are defined in the Laplace space. The quadrature formula of Stehfest [490] is implemented to transform the function values from the Laplace space to the original space in the time variable. A value of 5 or 6 is recommended to maximize accuracy of the approximation and to avoid numerical instabilities.

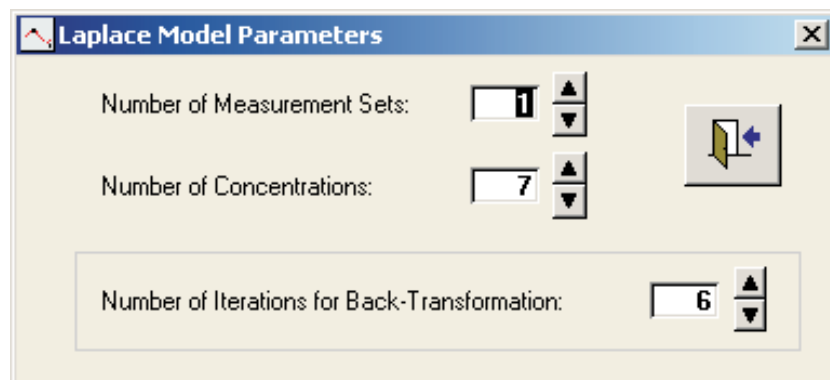


Figure 7.10: Parameters for Laplace Equations

7.2.4 Model Data for Ordinary Differential Equations

The numerical analysis program MODFIT is executed for models based on systems of ordinary differential equations. Also in this case, we need some integers that cannot be retrieved from the model function file or other data.

Number of Differential Equations: Define number of ordinary differential equations. The number of ODE's must coincide with the number of model functions for the right-hand side and the initial conditions on the Fortran or PCOMP input file.

Number of Measurement Sets: The number of measurement sets must coincide with the number of data sets as given in the input table for experimental data.

Number of Concentration Values: The number of concentrations must coincide with the number of concentrations as given in the input table for experimental data. If the value inserted is positive and the PCOMP input language is used, then a concentration variable must be declared in the model function file. If -1 is inserted, it is supposed that the fitting criteria depend on an additional concentration variable, and that one concentration value is assigned to each time value.

Shooting Index: Shooting technique can be introduced in the following situation:

1. There are as many measurement sets as differential equations.
2. Fitting criteria are the system variables of the differential equation in exactly the same order.
3. There are no additional constraints.
4. There are no zero weights (otherwise an artificial weight of 1.0E-7 is inserted).
5. There are no break points.

Integration is performed only from one shooting point to the next and then initialized with a shooting variable. The differences of shooting variables and solution at right-end of previous shooting interval lead to additional nonlinear equality constraints. The shooting index determines the number and position of shooting points:

- 0 - no shooting at all
- 1 - each measurement time is a shooting point
- 2 - every second measurement time is shooting point
- 3 - every third measurement time is shooting point

etc.

Method for Solving Ordinary Differential Equations: The numerical parameter estimation code MODFIT possesses interfaces to two subroutines for solving ordinary differential equations:

1. Implicit Radau-type Runge-Kutta code RADAU5 (Copyright ©2004, Ernst Hairer) of order 5 for stiff equations, confer Hairer and Wanner [199].
2. Runge-Kutta-Fehlberg method of order 4 to 5, see Shampine and Watts [470] with additional sensitivity analysis implemented by Benecke [36].

ODE Model Parameters

Number of Differential Equations: 2

Number of Measurement Sets: 2 Break Points

Number of Concentration Values: 0 Constraints

Shooting Index: 0

Integration Method: ☒ implicit ☐ explicit

Final Accuracy (absolute): 1E-06 Initial Step size: 0.0001

Final Accuracy (relative): 1E-06 Bandwidth of Jacobian: 0

Figure 7.11: ODE Parameters

Note that the first two codes use dense output, i.e., the integration is performed over the whole interval given by first and last time value, and the intermediate solution values are interpolated. In these cases, gradients are obtained by external numerical differentiation. If constant or variable break points are given, the integration is restarted at these time values with all initial tolerances supplied. Gradients are obtained by external numerical differentiation.

The explicit algorithm is capable to evaluate derivatives of the solution of the ODE internally, i.e., by analytical differentiation of the Runge-Kutta scheme. If constant or variable break points are defined, then the usage is prevented for some internal reasons.

Final Absolute Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the absolute global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. It is recommended to start with a relatively large accuracy, for

example 1.0E-6, together with a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Final Relative Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the relative global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. Again it is recommended to start with a relatively large accuracy, say 1.0E-6, and a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Initial Stepsize for Solving Differential Equations: Define initial stepsize for differential equation method used. The parameter is adapted rapidly by internal steplength calculation.

Bandwidth of Jacobian of Right-Hand Side: Implicit methods require the evaluation of the Jacobian of the right-hand side of an ordinary differential equation with respect to the system parameters, since they have to apply Newtons method to solve certain systems of nonlinear equations. The Jacobian is evaluated either numerically, by the automatic differentiation features of PCOMP, or must be provided by the user in form of Fortran statements. In any case, is possible that the Jacobian possesses a band structure depending on the ODE system. **EASY-FIT**^{*ModelDesign*} allows to define the bandwidth that is passed to the numerical integration routines to solve systems of internal nonlinear equations more efficiently. The bandwidth is the maximum number of non-zero entries below and above a diagonal entry of the Jacobian, and must be smaller than the number of differential equations. When inserting a zero value it is assumed that there is no band structure at all. The bandwidth can be defined only for the implicit integration routine.

Break Points: It is possible that the right-hand side of a system of ordinary differential equations is non-continuous with respect to integration time, e.g. if non-continuous input functions exist or if the model changes at certain time values. In case of constant break or switching points, respectively, the corresponding time values are to be defined in form of a separate table, where the integration is restarted with initial tolerances. The numerical values inserted, must vary between zero and the maximum experimental time value. Time values are ordered internally. If the table contains no entries at all, it is assumed that there are no constant break points with respect to the time variable.

On the other hand, it is possible that the last n_b optimization variables to be estimated, are used as break points in the code that defines the right-hand side and the initial conditions. In this case, the number n_b must be inserted. If $n_b > 0$ and constant break points exist in the corresponding table, then these values are ignored.

Constraints: Restrictions are allowed for models based on ordinary differential equations and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and the solution of the dynamical system at some of the given experimental time and concentration values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first

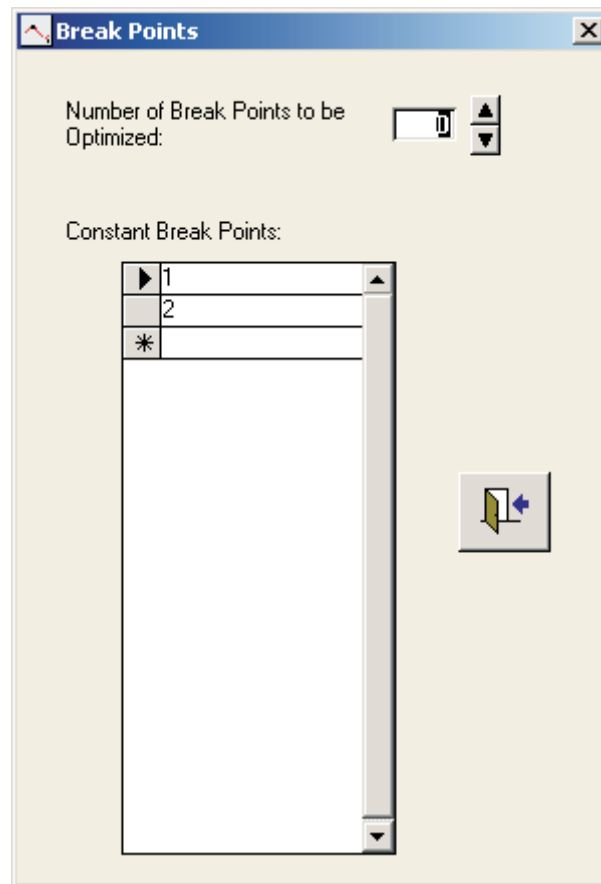


Figure 7.12: Break Points

in the input file for model functions. The table allows to define time and concentration values, for which solution values of the underlying dynamical system can be inserted:

- Order: Serial order number of constraints. Equality constraints must be defined first.
- Name: Arbitrary name for the constraint, to be printed in reports.
- Time: Corresponding time value at which a constraint is to be evaluated. Note that the time values are rounded to the nearest experimental time value, to avoid a rest of the integration. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.
- c/x-value: Corresponding concentration value at which a constraint is to be evaluated. Note that concentration values must be rounded to the nearest experimental concentration value. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental concentration value does not exist.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or concentration parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

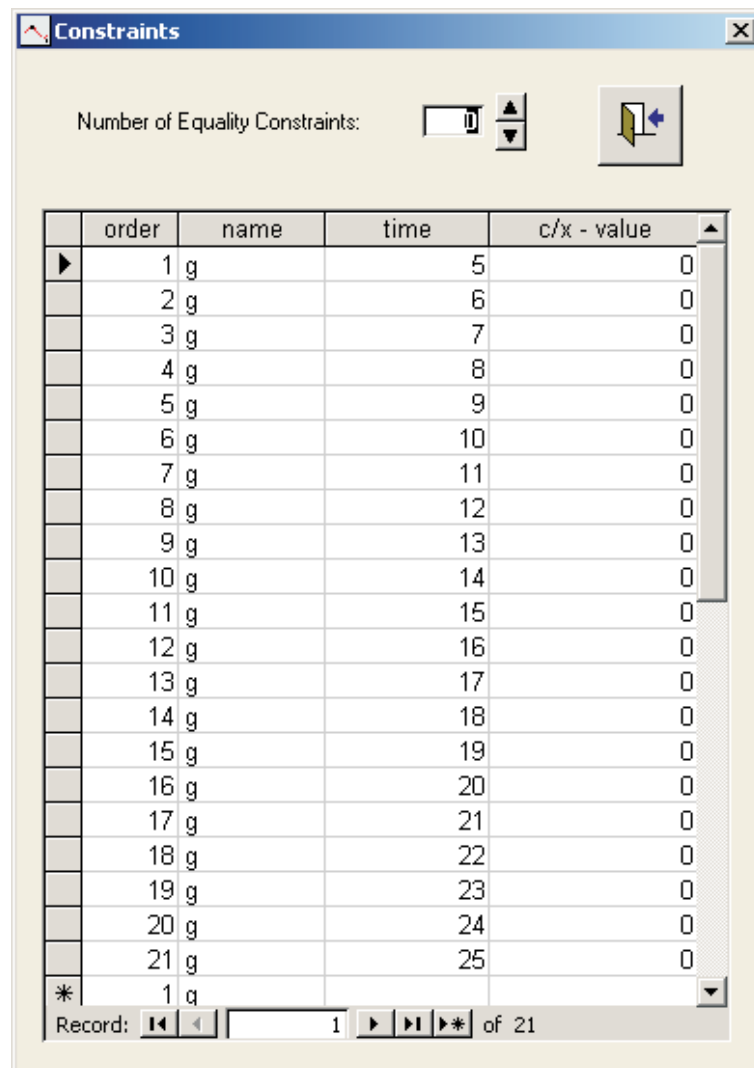


Figure 7.13: Constraints

7.2.5 Model Data for Differential Algebraic Equations

The code MODFIT solves parameter estimation problems based on differential algebraic equations, see Hairer and Wanner [199] for details. For the execution of MODFIT, we need some data that cannot be retrieved from the model function file or other sources.

Number of Differential Equations: Define total number of differential and algebraic equations. The number of DAE's must coincide with the number of model functions for the right-hand side and the number of initial conditions on the Fortran or PCOMP input file.

Number of Measurement Sets: The number of measurement sets must coincide with the number of data sets as given in the input table for experimental data.

Number of Concentration Values: The number of concentrations must coincide with the number of concentrations as given in the input table for experimental data. If the value inserted is positive and the PCOMP input language is used, then a concentration variable must be declared in the model function file. If -1 is inserted, it is supposed that the fitting criteria depend on an additional concentration variable, and that one concentration value is assigned to each time value.

Shooting Index: Shooting technique can be introduced in the following situation:

1. There are as many measurement sets as differential equations.
2. Fitting criteria are the system variables of the differential equation in exactly the same order.
3. There are no additional constraints.
4. There are no zero weights (otherwise an artificial weight of 1.0E-7 is inserted).
5. There are no break points.

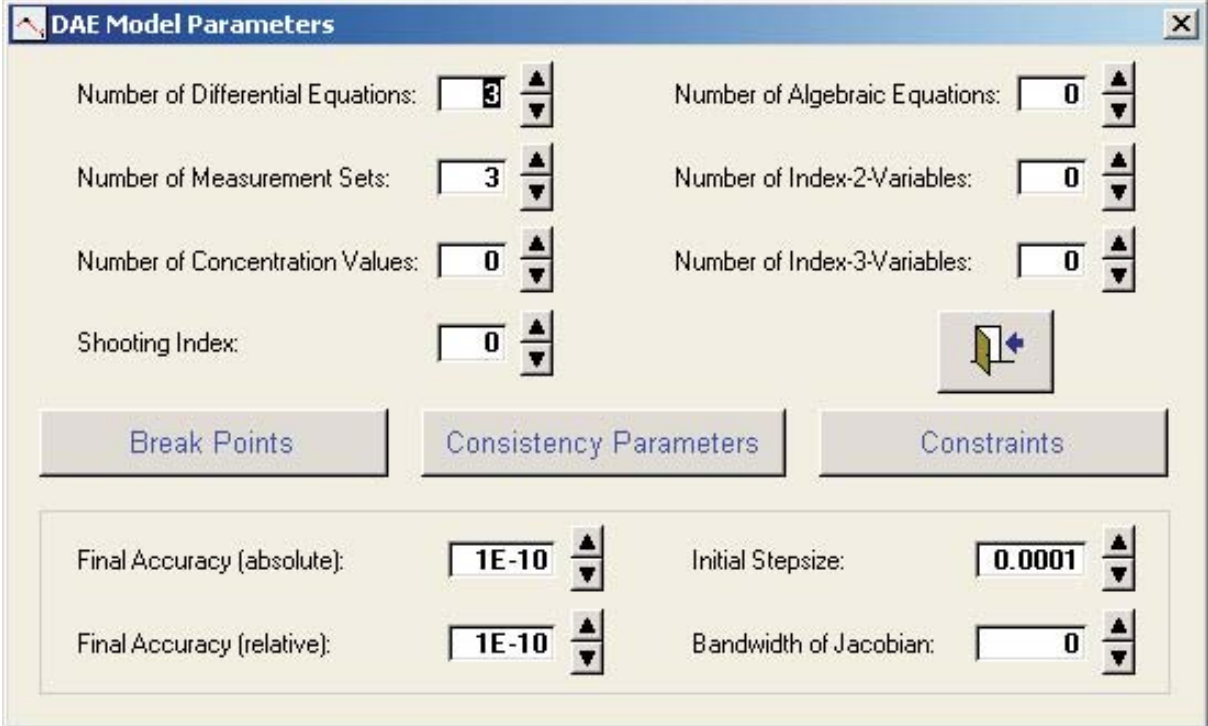
Integration is performed only from one shooting point to the next and then initialized with a shooting variable. The differences of shooting variables and solution at right-end of previous shooting interval lead to additional nonlinear equality constraints. The shooting index determines the number and position of shooting points:

- 0 - no shooting at all
- 1 - each measurement time is a shooting point
- 2 - every second measurement time is shooting point
- 3 - every third measurement time is shooting point

etc.

Number of Algebraic Equations: Define here the number of algebraic equations or algebraic variables, respectively. Note that the algebraic equations or algebraic variables, respectively, must follow the differential equations or differential variables, respectively, in the input file for model functions.

Number of Index-2-Variables: If a higher index system is given, define here the number of variables with index 2. The number is used to estimate the corresponding error and to scale



The image shows a software dialog box titled "DAE Model Parameters". It contains several input fields with up and down arrow buttons for adjustment. The fields are arranged in two columns. The first column includes "Number of Differential Equations" (set to 3), "Number of Measurement Sets" (set to 3), "Number of Concentration Values" (set to 0), and "Shooting Index" (set to 0). The second column includes "Number of Algebraic Equations" (set to 0), "Number of Index-2-Variables" (set to 0), and "Number of Index-3-Variables" (set to 0). Below these fields are three tabs: "Break Points", "Consistency Parameters", and "Constraints". At the bottom, there are four more input fields: "Final Accuracy (absolute)" (set to 1E-10), "Initial Stepsize" (set to 0.0001), "Final Accuracy (relative)" (set to 1E-10), and "Bandwidth of Jacobian" (set to 0).

Parameter	Value
Number of Differential Equations	3
Number of Algebraic Equations	0
Number of Measurement Sets	3
Number of Index-2-Variables	0
Number of Concentration Values	0
Number of Index-3-Variables	0
Shooting Index	0
Final Accuracy (absolute)	1E-10
Initial Stepsize	0.0001
Final Accuracy (relative)	1E-10
Bandwidth of Jacobian	0

Figure 7.14: DAE Parameters

the index-2-variables by the stepsize. The order of the DAE functions and variables is as follows:

1. index-1-variables
2. index-2-variables

Number of Index-3-Variables: If a higher index system is given, define here the number of algebraic variables with index 3. The number is only used for estimating the corresponding error and to scale the index-3-variables by the square of the stepsize. The order of the DAE functions and variables is as follows:

1. index-1-variables
2. index-2-variables
3. index-3-variables

An index- i -variable, $i = 1, 2, 3$, is defined by the number of differentiations of the variable needed to eliminate the algebraic variables and to formulate an equivalent system of ordinary differential equations.

Method for Solving Differential Algebraic Equations: The code MODFIT can be used for solving parameter estimation problems in differential algebraic equations, and executes an

implicit Runge-Kutta method of order 5 called RADAU5¹, see Hairer and Wanner [199]. Some parameters are to be set that cannot be retrieved from the model function file or other available data:

Final Absolute Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the absolute global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. It is recommended to start with a relatively large accuracy, e.g. 1.0E-6, together with a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Final Relative Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the relative global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. Again it is recommended to start with a relatively large accuracy, for instance 1.0E-6, and a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Initial Stepsize for Solving Differential Equations: Define initial stepsize for differential equation method used. The parameter is adapted rapidly by internal steplength calculation.

Bandwidth of Jacobian of Right-Hand Side: Implicit methods require the evaluation of the Jacobian of the right-hand side of a differential algebraic equation with respect to the system parameters, since they have to apply Newtons method to solve certain systems of nonlinear equations. The Jacobian is evaluated either numerically, by the automatic differentiation features of PCOMP, or must be provided by the user in form of Fortran statements. In any case, is possible that the Jacobian possesses a band structure depending on the DAE system. **EASY-FIT**^{ModelDesign} allows to define the bandwidth that is passed to the numerical integration routines to solve systems of internal nonlinear equations more efficiently. The bandwidth is the maximum number of non-zero entries below and above a diagonal entry of the Jacobian, and must be smaller than the number of differential equations. When inserting a zero value it is assumed that there is no band structure at all.

Break Points: Similar to ordinary differential equations, it is possible to define additional constant break or switching points, where the corresponding time values are to be defined in form of a separate table. The integration is restarted with initial tolerances at these time values. The numerical values inserted, must vary between zero and the maximum experimental time value. Time values are ordered internally. If the table contains no entries at all, it is assumed that there are no constant break points with respect to the time variable. On the other hand, it is possible that the last n_b optimization variables to be estimated, are used as break points in the code that defines the right-hand side and the initial conditions. In this case, the number n_b must be inserted. If $n_b > 0$ and constant break points exist in the corresponding table, then these values are ignored.

¹Copyright ©2004, Ernst Hairer

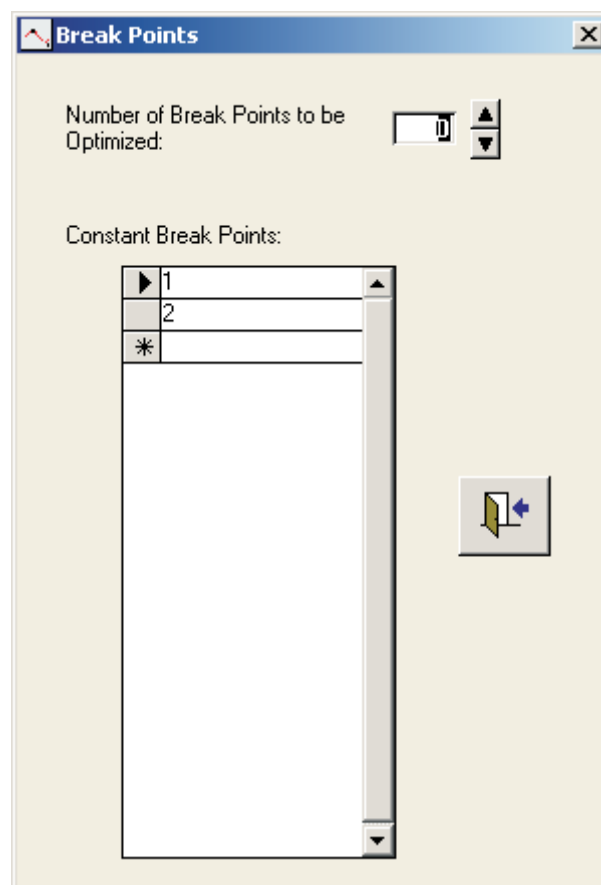


Figure 7.15: Break Points

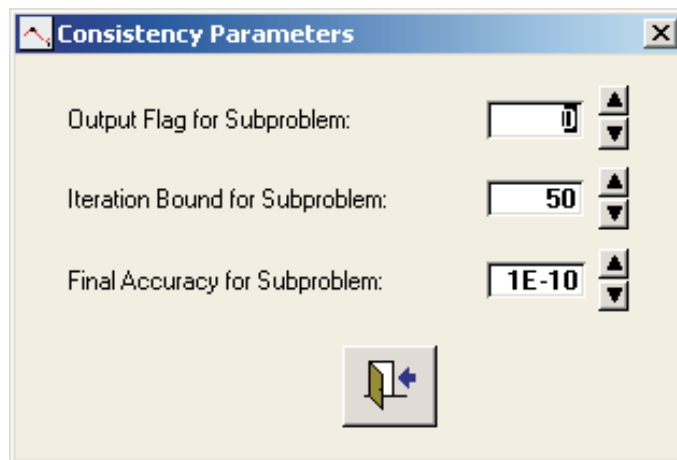


Figure 7.16: Consistency

Consistency Parameters: To achieve consistency of initial values, the corresponding system of algebraic equations is solved by the general purpose nonlinear programming method NLPQLP, see Schittkowski [427, 440]. For executing NLPQLP, a few parameters must be set:

Print flag: indicates the desired information to be displayed on the screen:

- 0 - no output at all
- 1 - only final summary of results
- 2 - one output line per iteration
- 3 - detailed output for each iteration

A value greater than 0 is only recommended in error cases to find out a possible reason for non-successful termination.

Maximum Number of Iterations: For computing consistent initial values, NLPQLP requires the maximum number of iterations to be defined here. A value of 50 is recommended.

Termination Accuracy: To compute consistent initial values by NLPQLP, one has to define the final accuracy by which the subproblem is solved. In case of exact derivatives, a value between 1.0E-8 and 1.0E-12 is recommended.

Constraints: Restrictions are allowed for models based on differential algebraic equations and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and the solution of the dynamical system at some of the given experimental time and concentration values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first in the input file for model functions. The table allows to define time and concentration values, for which solution values of the underlying dynamical system can be inserted:

- Order: Serial order number of constraints. Equality constraints must be defined first.
- Name: Arbitrary name for the constraint, to be printed in reports.
- Time: Corresponding time value at which a constraint is to be evaluated. Note that the time values are rounded to the nearest experimental time value, to avoid a rest of the integration of the system. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.
- c/x-value: Corresponding concentration value at which a constraint is to be evaluated. Note that concentration values must be rounded to the nearest experimental concentration value. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental concentration value does not exist.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or concentration parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

Constraints				
Number of Equality Constraints: <input type="text" value="0"/>				
	order	name	time	c/x - value
▶	1	g	5	0
	2	g	6	0
	3	g	7	0
	4	g	8	0
	5	g	9	0
	6	g	10	0
	7	g	11	0
	8	g	12	0
	9	g	13	0
	10	g	14	0
	11	g	15	0
	12	g	16	0
	13	g	17	0
	14	g	18	0
	15	g	19	0
	16	g	20	0
	17	g	21	0
	18	g	22	0
	19	g	23	0
	20	g	24	0
	21	g	25	0
*	1	q		
Record: <input type="text" value="1"/> of 21				

Figure 7.17: Constraints

7.2.6 Model Data for Time-Dependent Partial Differential Equations

For the execution of the numerical analysis program PDEFIT for models based on systems of partial differential equations, we need some data that cannot be retrieved from the model function file or other data:

Number of Differential Equations: Define number of partial differential equations. The number of PDE's must coincide with the number of model functions for the right-hand side on the Fortran or PCOMP input file.

Number of Integration Areas: Partial differential equations may be defined in different areas. Their total number is to be inserted here. The number must coincide with the number of areas defined in the table that defines the individual area structure. In each area, a uniform discretization grid is applied. Thus, different areas with smooth transitions can be used to change the grid size.

Starting Value for Spatial Interval: The spatial interval for the one-dimensional space coordinate is defined by the initial value to be defined here, and the individual lengths of the areas defined in the corresponding input table.

Order of Partial Differential Equation: If the right-hand side of the partial differential equation depends only on first spatial derivatives, it is not necessary to evaluate second order approximations for u_{xx} .

Flux in state equation: Flux functions facilitate the input of more complex equations and allow the application of special upwind or similar formulae in case of hyperbolic equations. If set to yes, there must be variables identifying flux functions and their spatial derivatives in the PCOMP input file, moreover a defining equation of the flux for each state variable.

Spatial Discretization: Partial differential equations are discretized with respect to the spatial variable by the following difference formulae:

- 3-point difference formula, recursively for second derivatives
- 5-point difference formula, recursively for second derivatives
- 5-point difference formula for first and second derivatives

These difference formulae except the third one, can individually be applied to the spatial derivatives of each state variable. In case of hyperbolic equations defined by flux functions, the following formulae are available:

- forward differences for first derivatives
- backward differences for first derivatives
- simple upwind formula
- second order scheme
- third order upwind-biased scheme
- ENO scheme

PDE Model Parameters

Number of Partial Differential Equations:

Number of Integration Areas:

Starting Value of Spatial Interval:

Order of spatial derivatives:

Flux in state equation:

Spatial Discretization:

- ☐ 3-pt formula, recursively for 2nd derivatives
- ☐ 5-pt formula, recursively for 2nd derivatives
- ☒ 5-pt formula for 1st and 2nd derivatives
- ☐ forward differences for first derivatives
- ☐ backward differences for first derivatives
- ☐ individual selection
- ☐ simple upwind formula (scalar)
- ☐ 2nd order scheme (scalar)
- ☐ 3rd order scheme
- ☐ ENO scheme (systems)

	Area	PDE	Name	Size	Lines	Status L	Status R	Discretization
▶	1	1	c _{im} (x,t)	80	31	0	0	0
	1	2	c _m (x,t)	0	0	2	2	0
*	1	1	area	1	11	1	1	1

Integration Method: ☒ implicit ☐ explicit

Accuracy (absolute): Initial Stepsize:

Accuracy (relative): Jacobian bandwidth:

ODE-Positions

Fit-Positions

Figure 7.18: PDE Parameters

The forward and backward schemes can also be used to discretize individual scalar equations out of a system of multiple equations, either only hyperbolic or mixed ones. However, the *wind* direction must be known a priori. In all other case, the upwind direction is determined internally. The simple upwind scheme and the second and third order schemes can be applied only to scalar equations. In case of the ENO scheme, an additional explicit Runge-Kutta methods is implemented to satisfy a so-called CFL stability condition.

Area Data for Partial Differential Equations: The structure the individual integration areas and therefore the positions of the transition equations are to be defined in form of a table. Note that only the status values are obligatory for each individual partial differential equation. It is sufficient to define the corresponding information only within one line that identifies the area. In this case, the equation number must be 1.

For each area and the differential equation identified by a serial number, the following data must be set:

Name:	The area may be characterized by an arbitrary string.
Size:	The length of the spatial interval of the corresponding area must be given.
Lines:	Within each area, an equidistant grid with respect to the spatial variable is used to transform the PDE into a system of ordinary equations. The number of grid points must be odd, to get an even number of equidistant intervals for applying Simpson's rule in case of numerical integration with respect to spatial variable. Note that the larger the number of grid points is, the larger is the resulting ODE.
Status:	<p>A status number identifies the type of the transition condition between areas, and also of the boundary conditions for each individual equation. The following options are allowed for the left and separately for the right end point of the area:</p> <p>0 - no boundary condition 1 - Dirichlet type boundary condition, i.e., function value given 2 - Neumann type boundary condition, i.e., derivative value given 3 - both types of boundary conditions</p>
Discretization:	<p>In case of addressing individual difference formulae to the state variables, the following approximation schemes can be combined:</p> <p>1 - 3-point difference formula, recursively for second derivatives 2 - 5-point difference formula, recursively for second derivatives 3 - forward differences for first derivatives 4 - backward differences for first derivatives</p>

Spatial Positions of Coupled ODE's: The positions of the spatial variable x where ordinary differential equations are coupled to the system of partial equations, must be given. The order must be increasing and any decimal value within the integration interval is allowed. The number and order of entries must coincide with the number and order of ODEs defined in the model function file. Decimal numbers are rounded to the nearest integer that describes a line of the discretized system.

Spatial Positions of Fitting Criteria: The positions of the spatial variable x where fitting criteria and corresponding measurement values are set, must be defined. The order must be increasing and any decimal value within the integration interval is allowed. The number and order of entries must coincide with the number and order of fitting functions of the model function file and, of course, also with the number of measurement sets given. Decimal numbers are rounded to the nearest integer that describes a line of the discretized system.

Methods for Solving Discretized ODE: The parameter estimation code PDEFIT possesses interfaces to several different subroutines for solving ordinary differential equations resulting from the discretization by the method of lines:

1. Explicit Runge-Kutta code DOPRI5 of order 4/5 based on Dormand and Prince formula, see Hairer, Nørsett and Wanner [197]. Steplength is adapted internally.
2. Implicit Radau-type Runge-Kutta code RADAU5 (Copyright ©2004, Ernst Hairer) of order 5 for stiff equations, see Hairer and Wanner [199].
3. Explicit Runge-Kutta method of order 4/5 with fixed stepsize for ENO discretization of PDE's, to satisfy the CFL stability condition. Usage is not recommended in case of non-homogeneous equations.

All codes use dense output, i.e., the integration is performed over the whole interval given by first and last time value, and intermediate solution values are interpolated. Gradients are obtained by external numerical differentiation.

Final Absolute Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the absolute global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. It is recommended to start with a relatively large accuracy, e.g. 1.0E-6, together with a low number of iterations, and to increase the accuracy when approaching a solution by restarts. The parameter is not needed for the explicit solver in case of an ENO scheme. In case of internal selection of a perturbation tolerance for computing numerical derivatives, however, the parameter serves as a guess for the accuracy by which function values are provided.

Final Relative Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the relative global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. Again it is recommended to start with a relatively large accuracy, say 1.0E-6, and a low number of iterations, and to increase the accuracy when approaching a solution by restarts. For the explicit solver with fixed steplength needed for ENO schemes, a reduction factor is to be specified by which the given stepsize is scaled if the CFL condition is not satisfied.

Initial Stepsize for Solving Differential Equations: Define initial stepsize for differential equation method used. The parameter is adapted rapidly by internal steplength calculation.

Bandwidth of Jacobian of Right-Hand Side: Implicit methods require the evaluation of the Jacobian of the right-hand side of the discretized system of ordinary differential equations with respect to the system parameters, since they have to apply Newtons method to solve certain systems of nonlinear equations. The Jacobian is evaluated either numerically, by the automatic differentiation features of PCOMP, or must be provided by the user in form of Fortran statements. In all cases, it is possible that the Jacobian possesses a band structure depending on the discretization scheme. **EASY-FIT**^{ModelDesign} allows to define the bandwidth that is passed to the numerical integration routines to solve systems of internal nonlinear equations more efficiently. The bandwidth is the maximum number of non-zero

entries below and above a diagonal entry of the Jacobian, and must be smaller than the total number of discretized differential equations. When inserting a zero value it is assumed that there is no band structure at all. The bandwidth can be defined only for the implicit integration routine.

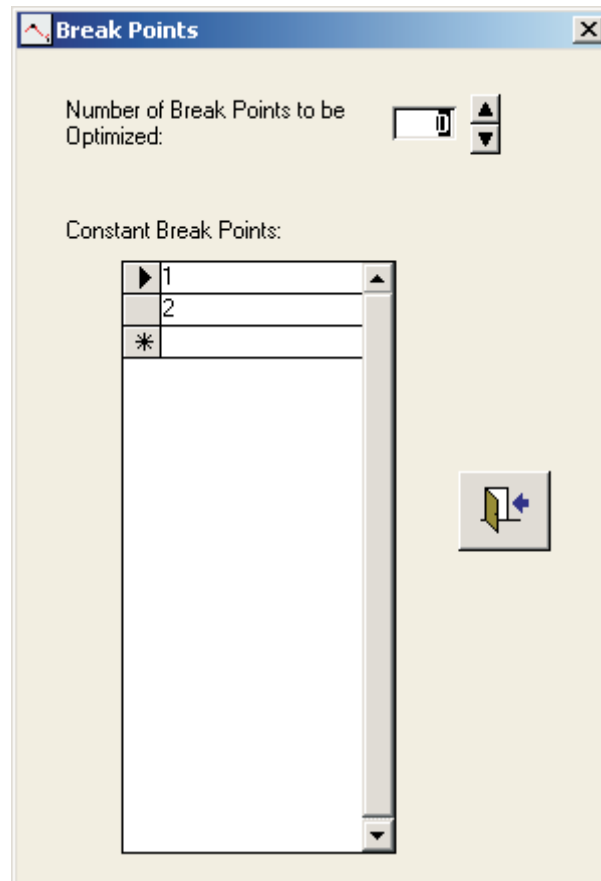


Figure 7.19: Break Points

Break Points: Similar to ordinary differential equations, it is possible to define additional constant break points, where the corresponding time values are to be defined in form of a separate table. The integration is restarted with initial tolerances at these switching values. The numerical values inserted, must vary between zero and the maximum experimental time value. Time values are ordered internally. If the table contains no entries at all, it is assumed that there are no constant break points with respect to the time variable.

On the other hand, it is possible that the last n_b optimization variables to be estimated, are used as break points in the code that defines the right-hand side and the initial conditions. In this case, the number n_b must be inserted. If $n_b > 0$ and constant break points exist in the corresponding table, then these values are ignored.

Constraints: Restrictions are allowed for models based on partial differential equations and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and the solution of the dynamical system at some of the given experimental time and spatial parameter values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first in the input file for model functions. The table allows to define time and spatial values, for which solution values of the underlying dynamical system can be inserted:

- Order: Serial order number of constraints. Equality constraints must be defined first.
- Name: Arbitrary name for the constraint, to be printed in reports.
- Time: Corresponding time value at which a constraint is to be evaluated.
Note that the time values are rounded to the nearest experimental time value, to avoid a re-integration of the system. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.
- c/x-value: Corresponding spatial parameter value at which a constraint is to be evaluated. The values are rounded to the nearest line number.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or spatial parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

Constraints				
Number of Equality Constraints: <input type="text" value="0"/>				
	order	name	time	c/x - value
▶	1	g	5	0
	2	g	6	0
	3	g	7	0
	4	g	8	0
	5	g	9	0
	6	g	10	0
	7	g	11	0
	8	g	12	0
	9	g	13	0
	10	g	14	0
	11	g	15	0
	12	g	16	0
	13	g	17	0
	14	g	18	0
	15	g	19	0
	16	g	20	0
	17	g	21	0
	18	g	22	0
	19	g	23	0
	20	g	24	0
	21	g	25	0
*	1	q		
Record: <input type="text" value="1"/> of 21				

Figure 7.20: Constraints

7.2.7 Model Data for Partial Differential Algebraic Equations

The numerical integration of systems of partial differential algebraic equations is very similar to the solution of PDE's without algebraic equations.

Number of Differential Equations: Define number of partial differential equations. The number of PDE's must coincide with the number of model functions for the right-hand side on the Fortran or PCOMP input file.

Number of Algebraic Equations: Define number of algebraic equations. The number of differential and algebraic equations must coincide with the number of model functions for the right-hand side on the Fortran or PCOMP input file.

PDAE Model Parameters

Number of Partial Differential Equations:

Number of Partial Algebraic Equations:

Number of Integration Areas:

Starting Value of Spatial Interval:

Order of spatial derivatives:

Flux in state equation:

Spatial Discretization

☐ 3-pt formula, recursively for 2nd derivatives ☐ forward differences for first derivatives

☒ 5-pt formula, recursively for 2nd derivatives ☐ backward differences for first derivatives

☐ 5-pt formula for 1st and 2nd derivatives ☐ individual selection

Area	PDAE	Name	Size	Lines	Status L	Status R	Discretization
▶ 1	1	u(x,t)	1	21	1	1	0
▶ 1	2	v(x,t)	1	21	1	1	0
* 1	1	area	1	11	1	1	1

ODE-Positions:

DAE-Positions:

Fit-Positions:

Consistency

Accuracy (absolute): Initial Stepsize:

Accuracy (relative): Jacobian bandwidth:

Figure 7.21: PDAE Parameters

Number of Integration Areas: Partial differential equations may be defined in different areas. Their total number is to be inserted here. The number must coincide with the number of areas defined in the table that defines the individual area structure. In each area, a uniform discretization grid is applied. Thus, different areas with smooth transitions can be used to change the grid size.

Starting Value for Spatial Interval: The spatial interval for the one-dimensional space coordinate is defined by the initial value to be defined here, and the individual lengths of the areas defined in the corresponding input table.

Order of Partial Differential Equation: If the right-hand side of the partial differential equation depends only on first spatial derivatives, it is not necessary to evaluate second order approximations for u_{xx} .

Flux in state equation: Flux functions facilitate the input of more complex equations. If set to yes, there must be variables identifying flux functions and their spatial derivatives in the PCOMP input file, moreover a defining equation of the flux for each state variable.

Spatial Discretization: Partial differential equations are discretized with respect to the spatial variable by the following difference formulae:

- 3-point difference formula, recursively for second derivatives
- 5-point difference formula, recursively for second derivatives
- 5-point difference formula for first and second derivatives
- forward differences for first derivatives
- backward differences for first derivatives

All of these difference formulae can individually be applied to the spatial derivatives of each state variable. The forward and backward schemes can also be used to discretize individual scalar equations out of a system of multiple equations, either only hyperbolic or mixed ones. However, the *wind* direction must be known a priori.

Area Data for Partial Differential Equations: The structure the individual integration areas and the positions of the transition equations are to be defined in form of a table. Note that only the status values are obligatory for each individual partial differential equation. It is sufficient to define the corresponding information only within one line that identifies the area. In this case, the equation number must be 1.

For each area and the differential equation identified by a serial number, the following data must be set:

Name:	The area may be characterized by an arbitrary string.
Size:	The length of the spatial interval of the corresponding area must be given.
Lines:	Within each area, an equidistant grid with respect to the spatial variable is used to transform the PDE into a system of ordinary equations. The number of grid points must be odd, to get an even number of equidistant intervals for applying Simpson's rule in case of numerical integration with respect to the spatial variable. Note that the larger the number of grid points is, the larger is the resulting ODE.
Status:	<p>A status number identifies the type of the transition condition between areas, and also of the boundary conditions for each individual equation. The following options are allowed for the left and separately for the right end point of the area:</p> <p>0 - no boundary condition 1 - Dirichlet type boundary condition, function value given 2 - Neumann type boundary condition, derivative value given 3 - both types of boundary conditions</p>
Discretization:	<p>In case of addressing individual difference formulae to the state variables, the following approximation schemes can be combined:</p> <p>1 - 3-point difference formula, recursively for second derivatives 2 - 5-point difference formula, recursively for second derivatives 3 - forward differences for first derivatives 4 - backward differences for first derivatives</p>

Spatial Positions of Coupled ODE's: The positions of the spatial variable x where ordinary differential equations are coupled to the system of partial equations, must be given. The order must be increasing and any decimal value within the integration interval is allowed. The number and order of entries must coincide with the number and order of ODE's defined in the model function file. Decimal numbers are rounded to the nearest integer that describes a line of the discretized system.

Spatial Positions of Coupled Algebraic Equations: The positions of the spatial variable x where algebraic equations are coupled to the system of partial equations, must be given. The order must be increasing and any decimal value within the integration interval is allowed. The number and order of entries must coincide with the number and order of algebraic equations defined in the model function file. Decimal numbers are rounded to the nearest integer that describes a line of the discretized system.

Spatial Positions of Fitting Criteria: The positions of the spatial variable x where fitting criteria and corresponding measurement values are set, must be defined. The order must be increasing and any decimal value within the integration interval is allowed. The number and order of entries must coincide with the number and order of fitting functions of the

model function file and, of course, also with the number of measurement sets given. Decimal numbers are rounded to the nearest integer that describes a line of the discretized system.

Method for Solving Discretized DAE: The resulting system of differential algebraic equations is solved by an implicit Runge-Kutta method of order 5 called RADAU5², see Hairer and Wanner [199]. The code is able to take algebraic equations into account, and use dense output, i.e., the integration is performed over the whole interval given by first and last time value, and intermediate solution values are interpolated. Gradients are obtained by external numerical differentiation.

Final Absolute Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the absolute global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. It is recommended to start with a relatively large accuracy, e.g. 1.0E-6, together with a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Final Relative Accuracy for Solving Differential Equations: Desired final accuracy for the differential equation solver with respect to the relative global error is to be inserted. In case of numerical approximation of gradients, the differential equation must be solved as accurately as possible. It is recommended to start with a relatively large accuracy, a low number of iterations, and to increase the accuracy when approaching a solution by restarts.

Initial Stepsize for Solving Differential Equations: Define initial stepsize for differential equation method used. The parameter is adapted rapidly by internal steplength calculation.

Bandwidth of Jacobian of Right-Hand Side: Implicit methods require the evaluation of the Jacobian of the right-hand side of the discretized system of ordinary differential equations with respect to the system parameters, since they have to apply Newtons method to solve certain systems of nonlinear equations. The Jacobian is evaluated either numerically, by the automatic differentiation features of PCOMP, or must be provided by the user in form of Fortran statements. In all cases, it is possible that the Jacobian possesses a band structure depending on the discretization scheme. **EASY-FIT**^{ModelDesign} allows to define the bandwidth that is passed to the numerical integration routines to solve systems of internal nonlinear equations more efficiently. The bandwidth is the maximum number of non-zero entries below and above a diagonal entry of the Jacobian, and must be smaller than the total number of discretized differential equations. When inserting a zero value it is assumed that there is no band structure at all.

Break Points: Similar to ordinary differential equations, it is possible to define additional constant break points, where the corresponding time values are to be defined in form of a separate table. The integration is restarted with initial tolerances at these switching values. The numerical values inserted, must vary between zero and the maximum experimental time

²Copyright ©2004, Ernst Hairer

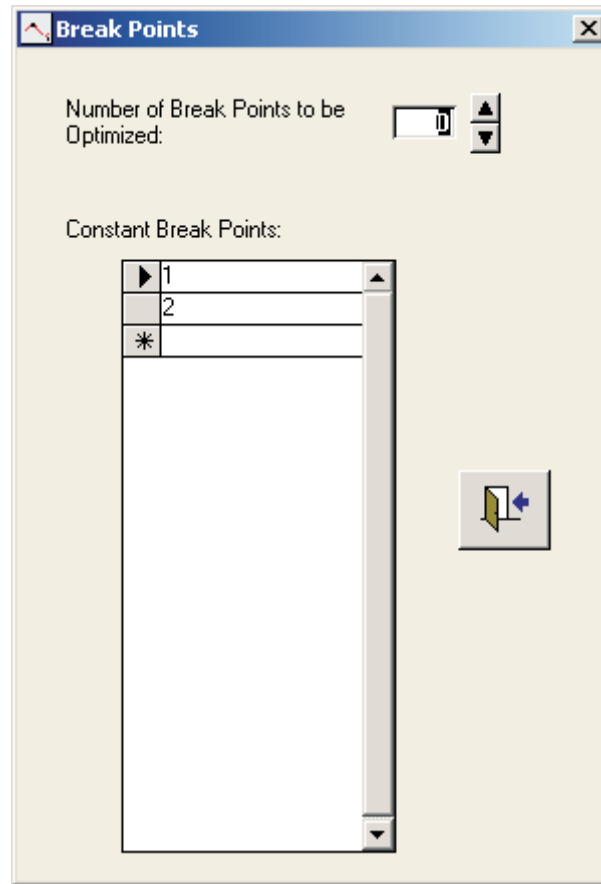


Figure 7.22: Break Points

value. Time values are ordered internally. If the table contains no entries at all, it is assumed that there are no constant break points with respect to the time variable.

On the other hand, it is possible that the last n_b optimization variables to be estimated, are used as break points in the code that defines the right-hand side and the initial conditions. In this case, the number n_b must be inserted. If $n_b > 0$ and constant break points exist in the corresponding table, then these values are ignored.

Consistency Parameters: To compute consistent initial values, the corresponding system of discretized algebraic equations is solved by the general purpose nonlinear programming method NLPQLP, see Schittkowski [427, 440]. For executing NLPQLP, a few parameters must be set:

Print flag: indicates the desired information to be displayed on the screen:

- 0 - no output at all
- 1 - only final summary of results
- 2 - one output line per iteration
- 3 - detailed output for each iteration

A value greater than 0 is only recommended in error cases to find out a possible reason for non-successful termination.

Maximum Number of Iterations: For computing consistent initial values, NLPQLP requires the maximum number of iterations to be defined here. A value of 50 is recommended.

Termination Accuracy: To compute consistent initial values by NLPQLP, one has to define the final accuracy by which the subproblem is solved. In case of exact derivatives, a value between 1.0E-8 and 1.0E-12 is recommended.

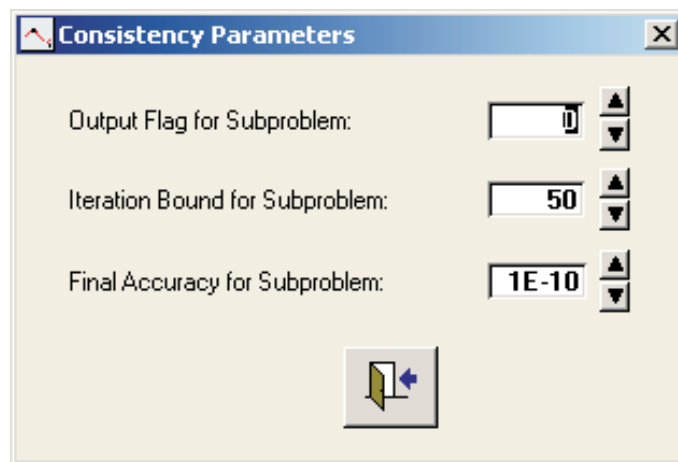


Figure 7.23: Consistency

Constraints: Restrictions are allowed for models based on partial differential equations and can be formulated in form of equality and inequality constraints with respect to the parameters to be optimized and the solution of the dynamical system at some of the given experimental time and spatial parameter values.

Where the total number of constraints can be retrieved from the subsequent table, the number of equality constraints must be supplied. Equality restrictions must be defined first in the input file for model functions. The table allows to define time and spatial values, for which solution values of the underlying dynamical system can be inserted:

Order: Serial order number of constraints. Equality constraints must be defined first.

Name: Arbitrary name for the constraint, to be printed in reports.

Time: Corresponding time value at which a constraint is to be evaluated. Note that the time values are rounded to the nearest experimental time value, to avoid a reset of the integration of the system. In case of doubt, insert dummy experimental data with zero weights, if constraints are to be defined at points for which an experimental time value does not exist.

c/x-value: Corresponding spatial parameter value at which a constraint is to be evaluated. The values are rounded to the nearest line number.

Note that equality restrictions must be defined first and that dummy values must be inserted for each constraint not depending on the time or spatial parameter. The number of lines in the table must coincide with the number of constraint functions defined on the model function input file (either Fortran or PCOMP).

Constraints
✕

Number of Equality Constraints:

▲

▼

	order	name	time	c/x - value
▶	1	g	5	0
	2	g	6	0
	3	g	7	0
	4	g	8	0
	5	g	9	0
	6	g	10	0
	7	g	11	0
	8	g	12	0
	9	g	13	0
	10	g	14	0
	11	g	15	0
	12	g	16	0
	13	g	17	0
	14	g	18	0
	15	g	19	0
	16	g	20	0
	17	g	21	0
	18	g	22	0
	19	g	23	0
	20	g	24	0
	21	g	25	0
*	1	q		

Record:

◀◀

◀

1

▶

▶▶

▶*

 of 21

Figure 7.24: Constraints

Chapter 8

Menu Commands

The menu commands of **EASY-FIT**^{*ModelDesign*} to define or alter data and functions, to start an optimization run or to get reports on numerical results, are described in this chapter.

8.1 File Command

By selecting either a name from the pick-list or by typing the name of an existing problem, an available data fitting problem of the database can be loaded. The pick-list can be sorted with respect to name, date, model type, or title either in ascending or descending order, to facilitate the access to problem data, see also the Save As command. New problems can be generated by the second option of the File command.

All problem data including the model function file can be copied from one problem to another within the actual database. **EASY-FIT**^{*ModelDesign*} needs the name of a destination problem for copying the actual problem to another one. If the desired problem exists in the database, its name is either taken from the pick-list or typed by hand. The pick-list can be sorted with respect to name, date, model type, or title either in ascending or descending order, to facilitate the access to problem data. Otherwise, a new problem is generated automatically, if the name defined is not found in the database. Note that all problem data including the model functions are copied, and that existing data are overwritten.

Another option of the File command is to import data from text-files with extensions DAT, FOR, and FUN, respectively. After defining the name of these files, a new problem with an arbitrary, user-provided name is generated. The format of the data file with extension DAT must coincide exactly with the format needed for the input of data for executing MODFIT.EXE or PDEFIT.EXE, respectively. The model function file with extension FUN or FOR, respectively, is the same edited by the user from the database directly.

Alternatively, it is possible to export problem data and to generate two text files in a directory specified by the user. A file with extension DAT contains all data in precisely the same input format as required by the numerical programs MODFIT.EXE and PDEFIT.EXE, to be able to execute these programs independently from **EASY-FIT**^{*ModelDesign*}. Another

EASY-FIT - [Main Form]

File Edit Start Report Data Delete Make Utilities ? Type a question for help

EASY-FIT ModelDesign Actual Problem: **SOIL** Prof. Klaus Schittkowski Bayreuth, Germany Version: 5.00 (Jan 2009)

Model Data Experimental Data

Model: **PDE**
 Information: Diffusion of water through soil, convection and dispersion
 Project Number: Demo Unit for X-Values: t
 User Name: Schittkowski Unit for Y-Values: Names
 Measurement Set: Experimental Unit for Z-Values: x
 Date: 14.12.1998
 Memo: van Genuchten M.T., Wierenga P.J. (1976): Mass transfer studies in sorbing porous media. 1. Analytical solutions, Soil Sci. Soc. Am. Journal, Vol. 44, 892-898
 Anderson F., Olsson B. (eds.) (1985): Lake Gaadsjön. An acid forest lake and its catchment, Ecological Bulletins, Vol. 37

Model Type:

- explicit** - explicit model functions with concentration values
- system** - system of steady state equations
- Laplace** - Laplace formulation of model functions with concentration values and internal backtransformation
- ODE** - system of ordinary differential equations with initial values, concentration values, and switching points
- DAE** - system of differential algebraic equations up to index 3 with initial values, concentration values, and switching points
- PDE** - system of one-dimensional, time-dependent partial differential equations with initial and boundary values, disjoint integration areas, coupled ODE's, flux functions and switching points
- PDAE** - same as for PDE, but with additional algebraic partial differential equations and coupled DAE's

Parameters to be Estimated:

no	name	lower bound	starting value	upper bound	optimal	PL
1	pm	1.00E-03	1.0000E+00	1.00E+04	1.1438E+01	3
2	pim	1.00E-05	1.0000E+00	1.00E+05	6.0377E+00	2
3	Dm	1.00E-05	1.0000E+02	1.00E+05	3.7099E+02	1
*		0.00E+00	1.0000E+00	1.00E+05		

Update Simulation Data Fitting Experimental Design

Model Parameters: Language: **PCOMP** **FORTTRAN**
 Norm: Sum of Absolute Residual Values
 Sum of Squared Residuals
 Maximum of Residual Values

Record: 1173 of 1398
 Model name (optional)

Figure 8.1: Main Form

file with extension FUN or FOR contains the model functions either in the PCOMP input format or in form of a FORTRAN subroutine as specified by the user. Possible reasons for exporting data are the execution of the numerical codes outside of the database or the possibility to copy problem data to another system. Note that these text files can be imported again by as outlined

The File command also allows to define a filter for selecting a subset of parameter estimation problems from the database. The search mask contains the following items:

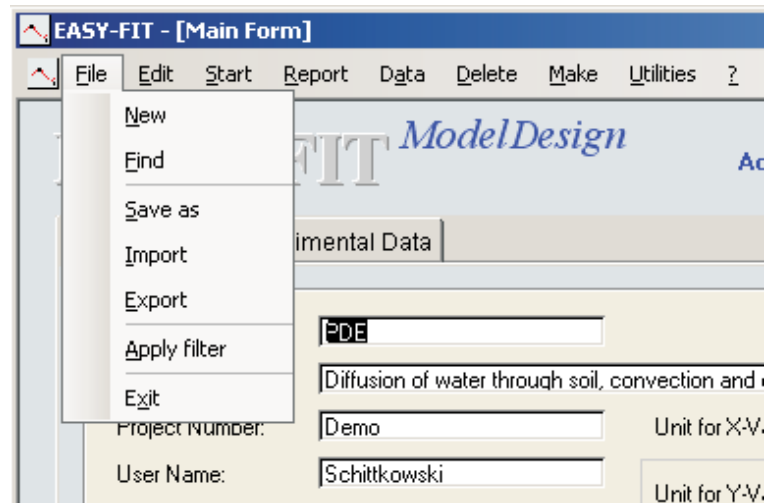


Figure 8.2: File Command

Problem Identifier
 Model Name
 Information
 Project Number
 User Name
 Measurement Set
 Date

The first six strings may contain a '*' for determining a group of problems. If a date is defined, then all problems are selected with date less than or equal to the given one. The corresponding database query assumes that at least some of the input fields contain non-empty strings.

Finally, the filtered problems can be sorted subject to the same input fields either in ascending or descending order.

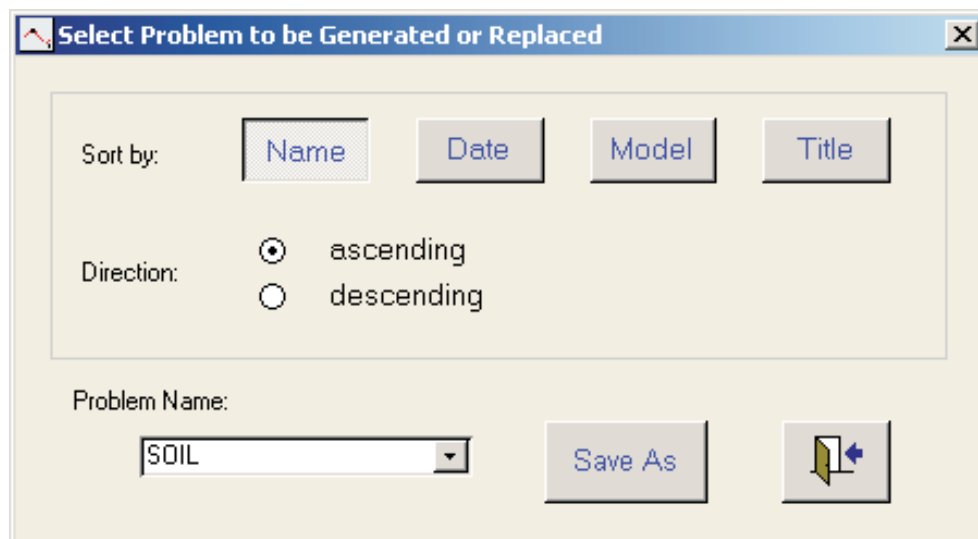


Figure 8.3: Save As Command

8.2 Edit Command

All data that define the dynamical model can be defined and changed by moving to the corresponding input field directly or by pushing the corresponding button. Alternatively, the Edit command allows to go directly to the corresponding area, i.e., an input field, a subtable, a subform or a file to be edited.

EASY-FIT^{ModelDesign} is delivered with two editors, an internal form ([EASY-FIT]) and an external executable file with syntax highlighting (EDITOR.EXE). Both allow direct parsing of PCOMP code or compilation and link of Fortran code. In case of PCOMP input, the corresponding parser can be started directly by an editor command. Otherwise, Fortran code must be generated, but a direct compilation and link is also possible. The order by which variables and functions are to be inserted, is predetermined by the underlying model structure.

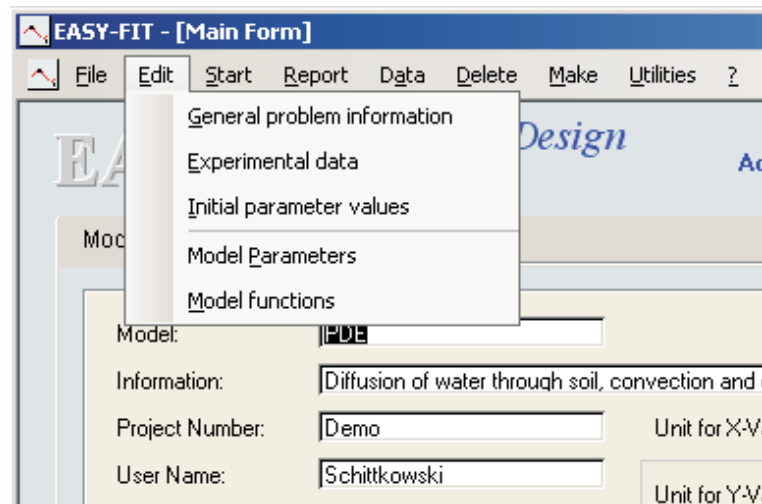


Figure 8.4: Edit Command

8.3 Start Command

Depending on the mathematical model type, **EASY-FIT**^{ModelDesign} starts one of the numerical codes MODFIT or PDEFIT. The codes perform either a simulation at a given parameter set or start an optimization cycle. **EASY-FIT**^{ModelDesign} generates a suitable input file with all data and tolerances required by the numerical algorithm. After termination, the results are read in and stored in the database.

In case of an initial analysis, a simulation is performed with respect to the given set of parameters as provided by the user, or the data fitting iteration cycle is started from these parameters. Otherwise, a restart is performed, i.e., the simulation or optimization run started with parameters from the database that are calculated by a previous run. In this case, the user is asked whether these values should replace the existing parameter values or not.

Note that the numerical results are sent to the database only if the displayed window is not closed before the numerical code terminated completely and an information message is visible.

If a PCOMP input file for declaring model functions is chosen, then the statements can be parsed directly from the corresponding form. If, on the other hand, an external editor is preferred, then the input cannot be parsed directly and the last line of the Start menu offers the corresponding command to execute the parser.

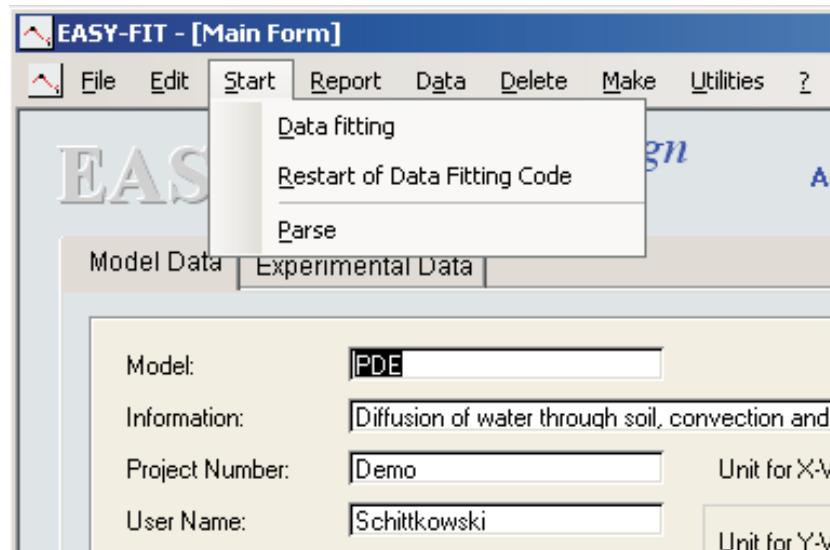


Figure 8.5: Start Command

8.4 Report Command

The report command serves to produce reports and function and data plot. A text report is generated directly from the database, and is displayed on screen. Reports contain the following information:

- General information about the parameter estimation problem, the data and the model, as provided by the user.
- Some numerical problem data, for example number of differential equations, number of measurement sets etc.
- User defined tolerances for the parameter estimation algorithm and the subproblem solver.
- Numerical performance results, i.e., number of function evaluations and final residual.
- Optimization variables as computed by the algorithm, together with starting values and bounds.
- Parameter estimation data, i.e., time, concentration, measurement, and model values, also all weights, listed separately for each set of experimental data.
- Constraint values, if restrictions exist.

In addition, we display for each individual set of experimental data a summary of some characterizing statistical parameters,

DOF	-	degrees of freedom
MV	-	mean value of experimental data
SOS	-	sum of squares
RSOS	-	relative sum of squares (square root of SOS divided by DOF)
MR	-	mean value of residuals (absolute values)
GOF	-	goodness of fit (one minus SOS/sum of squared differences of MV and model values)

Especially the goodness of fit value serves as a valuable scaling-invariant measure for comparing residuals. These data are only displayed if the number of experiments per data set is higher than the number of variables.

If a simulation run was performed with a positive significance level, then statistical error analysis data can be displayed on request. They contain the following information:

- Variance/covariance matrix
- Correlation matrix

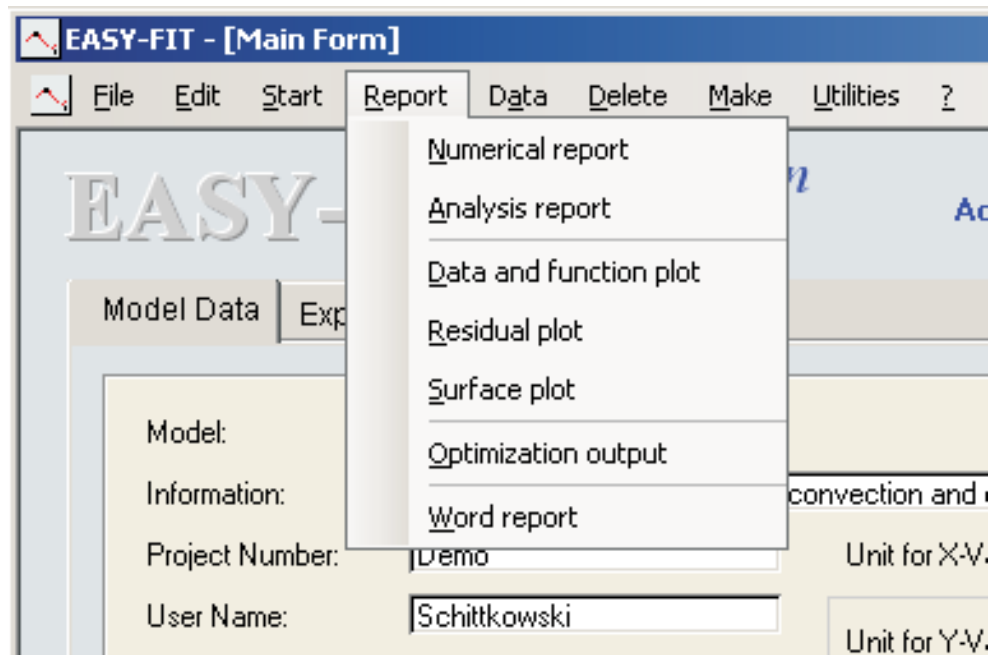


Figure 8.6: Report Command

- Estimated variance of residuals
- Confidence intervals for parameters

Confidence intervals of estimated parameters are computed for the significance levels 1%, 5% or 10%, respectively. These data may help to distinguish between relevant and redundant parameters and to get an impression about the quality of the model and experimental data.

Moreover, graphical plots show the fitting functions together with the experimental data given, the individual residuals and in addition three-dimensional plots. 3D-plots are generated only if concentration values are defined or in case of partial differential equations, where the numerical solution functions of the system is displayed. In both cases, three-dimensional surface plots are available.

Plots of model functions and experimental data are either generated by the internal plot program of **EASY-FIT**^{ModelDesign}, or for the external graphics system GNUPLOT¹. The standard plot program is implemented in form of a separate Fortran program called SP_PLOT.EXE. Plot data generated by MODFIT or PDEFIT are passed directly to SP_PLOT and GNUPLOT on files and are not kept in the database. A user has the option to require also an overlay of function and data plots. Three dimensional plots can be viewed from different angles.

¹Copyright ©1986-1993,1998,2004, Thomas Williams, Colin Kelley

Plots can be generated for the public domain software GNUPLOT, that must reside in a directory. The corresponding command file to start the program, is called GNUPLOT.GNU. Optionally, this file can be edited before starting GNUPLOT, to modify and adapt plot commands. It is possible to change the display style or to start a printout. To get the corresponding pop-up menu, one has to press the right mouse button. It is very important to close the plot correctly by answering the GNUPLOT-request correctly. Otherwise, the system may break down when trying to get the next plot. Plot data are read in directly from a text file.

For problems with at most one concentration value or at most one time value, two-dimensional plots are generated that show the experimental data and the model function values within the interval determined by the first and last time or concentration value, respectively. In the first case, measurement and model values are displayed over time, in the second case over concentration.

If more than one time and more than one concentration value is given or if partial differential equations are integrated, then a three-dimensional surface plot is generated where the time variable corresponds to the first horizontal x-axis, the concentration or spatial variable to the second horizontal z-axis and the function values to the vertical y-axis. Surface plots show either the fitting function in case of concentrations, or the model function, i.e., the solution function of the partial differential equation in the other case.

In case of more than one measurement set, plots are repeated for each set and are displayed in the order in which the data sets are defined. Overlays are admitted for the internal graphics system and for GNUPLOT, where the number of overlays is determined by the user.

Residual plots are introduced to detect visually systematic deviations of experimental data from the model function. The plots show the individual deviations in form of two-dimensional graphs. The horizontal axis displays the corresponding serial number, not the actual time or concentration value, respectively. In case of several measurement sets, the plots are repeated for each set and are displayed in the order in which the data sets are defined.

The original output of the selected least squares algorithm is directed to a file depending on the chosen print level. Subsequently the output can be displayed on request. However, one has to be a bit familiar with the underlying mathematical theory to understand the data in detail. For the meaning of the parameters displayed, it is necessary to read the corresponding user guides. If the internal editor is used and if MODFIT or PDEFIT generate too much output, it is possible that the size of the text size extends 32 K. To avoid an internal error situation, please switch to another editor in this case, for example to EDITOR.EXE.

8.5 Data Command

The menu command offers a few flexible opportunities for im- and export of measurement data.

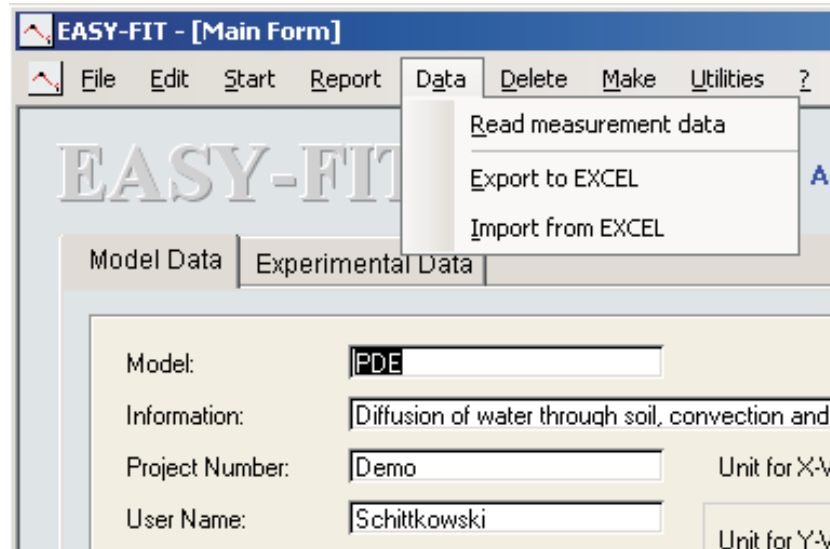


Figure 8.7: Import of Experimental Data

Experimental data can be read from any text file in standard format, where the corresponding time, concentration, measurement, and weight data must be organized in exactly the same order in which they are used in the input table of **EASY-FIT**^{ModelDesign}. There is no special format required for real numbers in the input file, but decimal or exponential numbers are not to be separated by anything else than a blank or new line.

Alternatively, data may be organized in rows possessing identical structure, where the initial column position and the length of an item to be read must be known in advance. These data are to be inserted by the user. In case of different concentration values, then each subset of rows belonging to one concentration, has the same structure, i.e., order and column positions of the data to be read.

All existing experimental data are deleted before reading new ones. The data input stops as soon as the end of file is reached.

Measurement data can be exported to an EXCEL file with extension XLS. Full path must be selected, where the default file name can be changed. For each set of measurement data, a corresponding EXCEL column is generated. In addition to time, concentration, experimental, and weight values, also model function values, residuals, and relative errors are exported together with column headings in the first row of the spreadsheet. These data can be used for example to plot data and results. After exporting data, EXCEL can be launched directly.

Directory: C:\TEMP\ ... browse

Extension: .DAT

Existing Files: C:\TEMP\SEQ_EXP.DAT
C:\TEMP\SEXP.DAT

File Name: C:\TEMP\SEQ_EXP.DAT

HINT: To select a file from the list, please click on this file name, then on the field.

Import Data in Standard Form

Import Data from Selected Columns

Time Values: 1
Concentration Values: 28

Data Set	Value	Length	Weight	Length	Data Set	Value	Length	Weight	Length	Data Set	Value	Length	Value	Length
1	10	7	0	0	11	0	0	0	0	21	0	0	0	0
2	0	0	0	0	12	0	0	0	0	22	0	0	0	0
3	0	0	0	0	13	0	0	0	0	23	0	0	0	0
4	0	0	0	0	14	0	0	0	0	24	0	0	0	0
5	0	0	0	0	15	0	0	0	0	25	0	0	0	0
6	0	0	0	0	16	0	0	0	0	26	0	0	0	0
7	0	0	0	0	17	0	0	0	0	27	0	0	0	0
8	0	0	0	0	18	0	0	0	0	28	0	0	0	0
9	0	0	0	0	19	0	0	0	0	29	0	0	0	0
10	0	0	0	0	20	0	0	0	0	30	0	0	0	0

Figure 8.8: Import of Experimental Data

Experimental data can directly be imported from an EXCEL spreadsheet, i.e., a file with extension XLS. For each set of measurement data, i.e., time, concentration, measurements, and weights, a cell range must be defined. The range should not cover more than one spreadsheet column. The order of concentration data must be the same as required by **EASY-FIT** *ModelDesign*. Time values are required in any case, and all existing experimental data are deleted before reading new ones.

Note that EXCEL Spreadsheet Version 97 is supported.

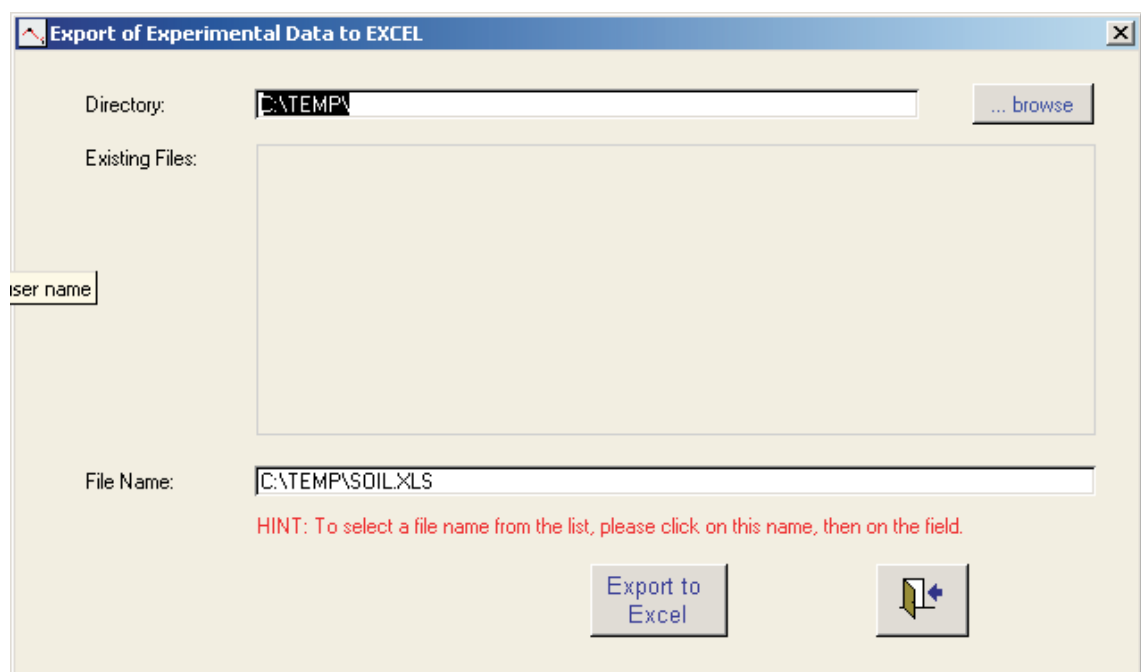


Figure 8.9: Export of Experimental Data to EXCEL

Import of EXCEL Data

Directory:

Existing Files:

File Name:

HINT: To select a file from the list, please click on this file name, then on the field.

Time Values:

Concentration Values:

start	end
A2	A767
E2	E767

Data Set	Values		Weights	
	start	end	start	end
1	B2	B767		
2	:	:	:	:
3	:	:	:	:
4	:	:	:	:
5	:	:	:	:
6	:	:	:	:
7	:	:	:	:
8	:	:	:	:
9	:	:	:	:
10	:	:	:	:

Data Set	Values		Weights	
	start	end	start	end
11	:	:	:	:
12	:	:	:	:
13	:	:	:	:
14	:	:	:	:
15	:	:	:	:
16	:	:	:	:
17	:	:	:	:
18	:	:	:	:
19	:	:	:	:
20	:	:	:	:

Data Set	Values		Weights	
	start	end	start	end
21	:	:	:	:
22	:	:	:	:
23	:	:	:	:
24	:	:	:	:
25	:	:	:	:
26	:	:	:	:
27	:	:	:	:
28	:	:	:	:
29	:	:	:	:
30	:	:	:	:

Figure 8.10: Import of Experimental Data from EXCEL

13

8.6 Delete Command

The actual problem is deleted from the database. All corresponding problem files, especially the model function file with extension <name>.FUN or <name>.FOR, are scratched from the problem directory of **EASY-FIT**^{ModelDesign}.

Alternatively, a search mask may be defined by the user to delete a complete subset of parameter estimation problems from the database. The search mask contains the following items:

Problem Identifier
Model Name
Project Number
User Name
Measurement Set
Date

The first five strings may contain a '*' for determining a group of problems. If a date is defined, then all problems are deleted with date less than or equal to the given one. Moreover, it is possible to require confirmation of deletion of each individual problem.

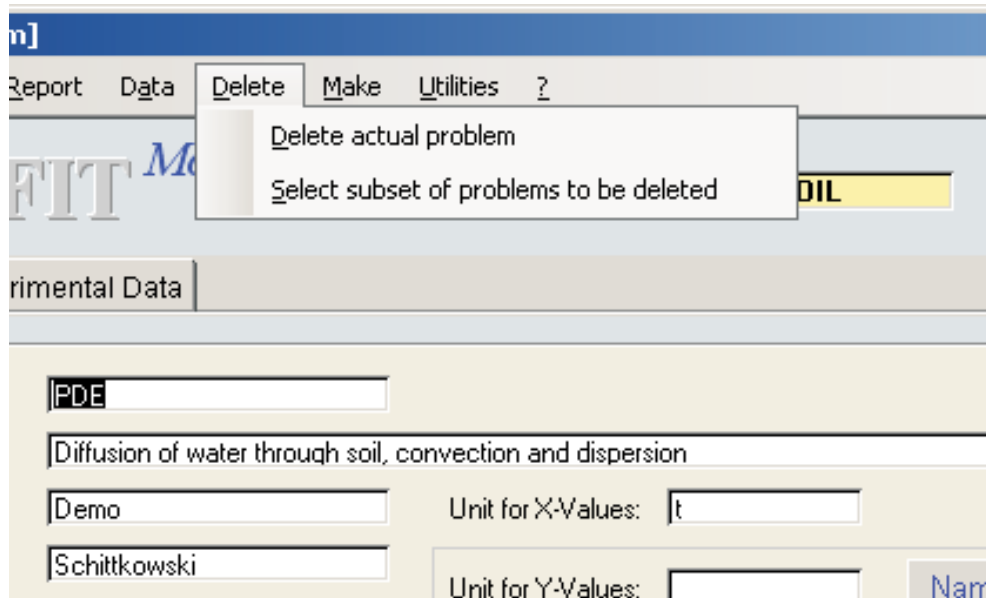


Figure 8.11: Delete Command

8.7 Make Command

If model functions are implemented in form of a Fortran code, the command allows to compile and generate executable code. The corresponding compiler and linker calls are adapted through the Utilities command, and must be contained in two DOS batch files with names COMPILE.BAT and LINKER.BAT.

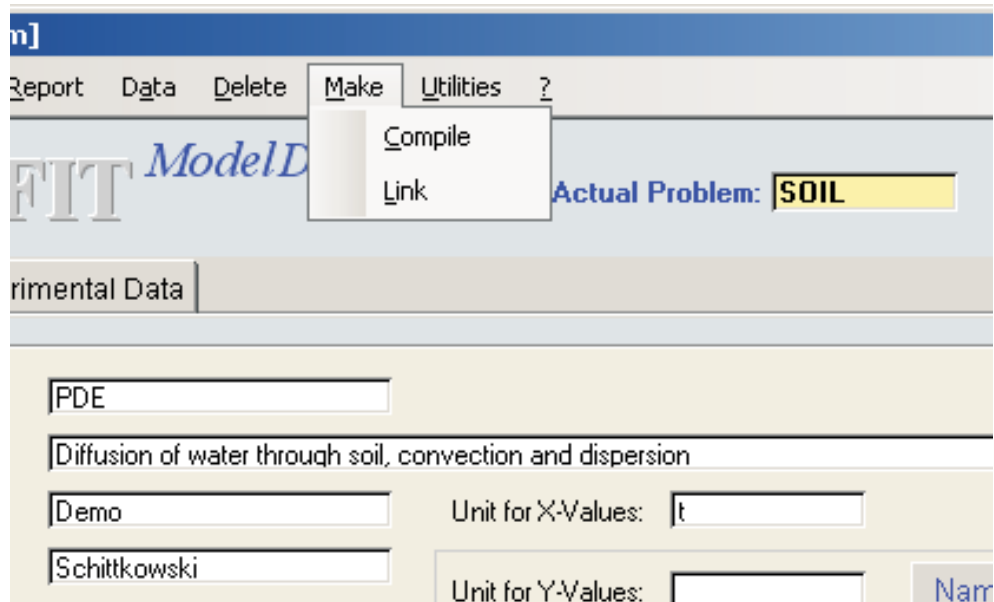


Figure 8.12: Make Command

8.8 Utilities Command

Through a couple of menu items **EASY-FIT**^{ModelDesign} can be adapted to individual situations. The following subcommands are available:

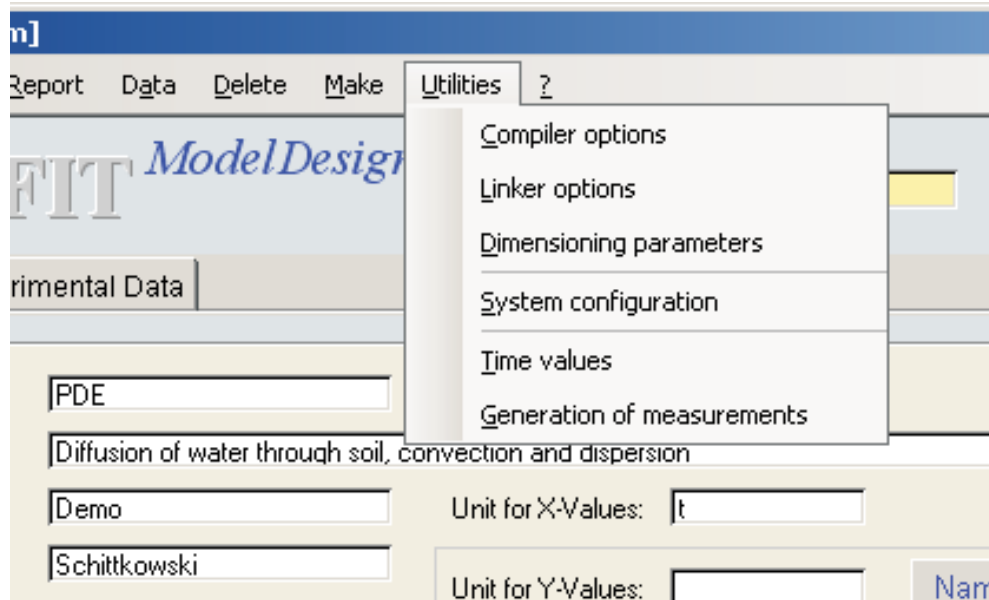


Figure 8.13: Utilities Command

Compiler Options: A DOS batch file with name COMPILE.BAT must contain all necessary compiler options. Default execution commands for various Fortran compilers are included, and can be modified in the editor window displayed.

Linker Options: Similar to the compiler, also all linker options can be adapted and reset by a user. The file to be edited, is called LINKER.BAT. The object codes required to link the complete program, are set automatically by **EASY-FIT**^{ModelDesign}.

Dimensioning Parameters: Parameter estimation problems can differ in their size dramatically. In some cases, we have an extremely large number of experimental data, in some others a large number of variables or system functions. It is not reasonable to compile and link numerical codes with extraordinary large dimensioning parameters to serve all possible extreme situations. Thus, the dimensioning parameters can be adapted. When activating the menu item, an include file with all dimensioning parameters is opened for editing. The meaning of the parameters is explained by initial comments, also the default values can be retrieved from that file. The user has the option compile and link FORTRAN codes directly from the corresponding form. The include file can be saved under a separate name or on a separate directory.

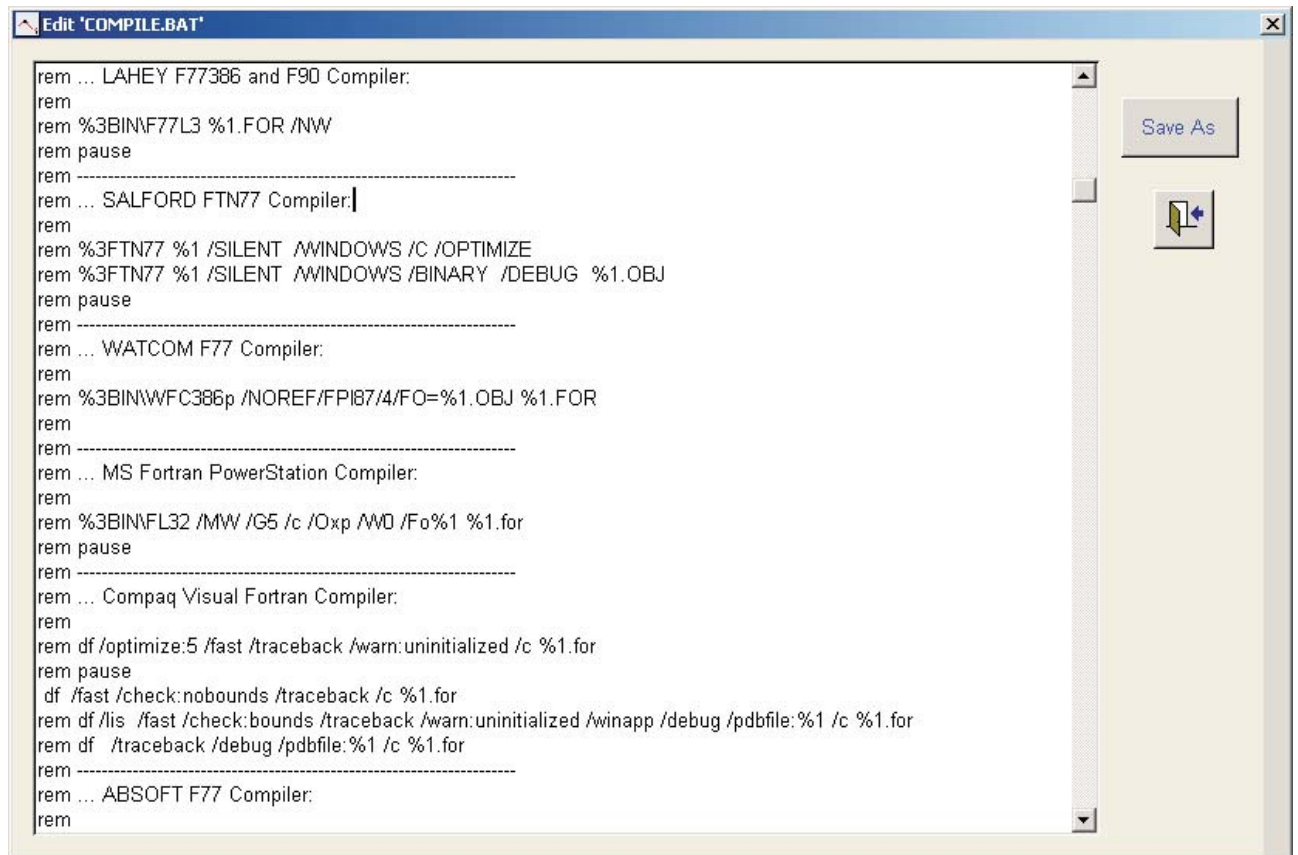


Figure 8.14: Compiler Options

System Configuration: **EASY-FIT**^{ModelDesign} needs to know where to find some files. Thus, a couple of directory names must be set according to the special needs of the user and depending on the special environment. Also an alternative Windows editor may be defined to be used for the input of model functions. It is recommended to check the available strings immediately during the first installation of **EASY-FIT**^{ModelDesign}. The actual version of **EASY-FIT**^{ModelDesign} possesses interfaces for the Watcom F77/386, the Salford FTN77, the Lahey F77L-EM/32, the Compaq Visual Fortran, Absoft Pro Fortran, the Microsoft Fortran PowerStation, and the Intel Visual Fortran compilers. The corresponding compiler name, some path names and the execution commands for compiler and linker are set in a configuration form.

Moreover, the graphics system may be changed, see the description of the Report command for more information. In addition, the form contains flags for editing GNUPLOT commands before executing the plot program and for allowing overlay of plots in case of using the standard graphics. A default path may be set that is inserted into the corresponding

form when looking for files for importing or exporting numerical data and model functions.

Generation of Time Values: The generation of equidistant or exponential time values might be the first step to generate a data fitting problem. Also for investigating stability or identifiability of a certain model before inserting experimental data, it might be desirable to perform some preliminary tests with artificial measurement values. The user has to provide the desired type of distribution, the final time, and the number of time values to be generated. Note that the zero time is always generated and not counted. If n is the number of time points until T , equidistant time values are $t_i = T \frac{i}{n}$ and exponential time value with shift s are given by the formula

$$t_i = T \frac{\exp(s \frac{i}{n}) - 1}{\exp(s) - 1}$$

for $i = 1, \dots, n$.

Generating of Measurements: When investigating stability or identifiability of a certain model before inserting experimental data, it might be desirable to perform some preliminary tests with artificial measurement values. Proceeding from a previously performed simulation run, it is possible to let the simulation results be inserted in the table containing the parameter estimation data. Subsequently, the given starting values for parameters to be estimated, can be disturbed and a parameter estimation run can be started from this initial point.

Note that random errors should be added to the generated measurements to simulate experimental real-life situations. To generate simulated measurement data, one has to define time values and, if necessary, also concentration values, furthermore zero entries in the column that has to get the simulated measurement values, and corresponding weights.

In many cases, it might be desirable to add some randomly generated errors to available measurement values. A typical situation arises when testing the identifiability of parameters. In this case, one would generate some artificial measurements by a simulation run at predetermined parameter values. A subsequent random perturbation and a data fitting test run from some other starting values indicates whether parameters can be identified or not. There are two possibilities:

a) Uniform distribution: The user is asked to specify the percentage of a relative error that is to be added to the experimental value. More precisely, if an error of $t\%$ is to be generated, then each value, say y_i , is replaced by

$$y_i(1 + t(2r_i - 1)/100) .$$

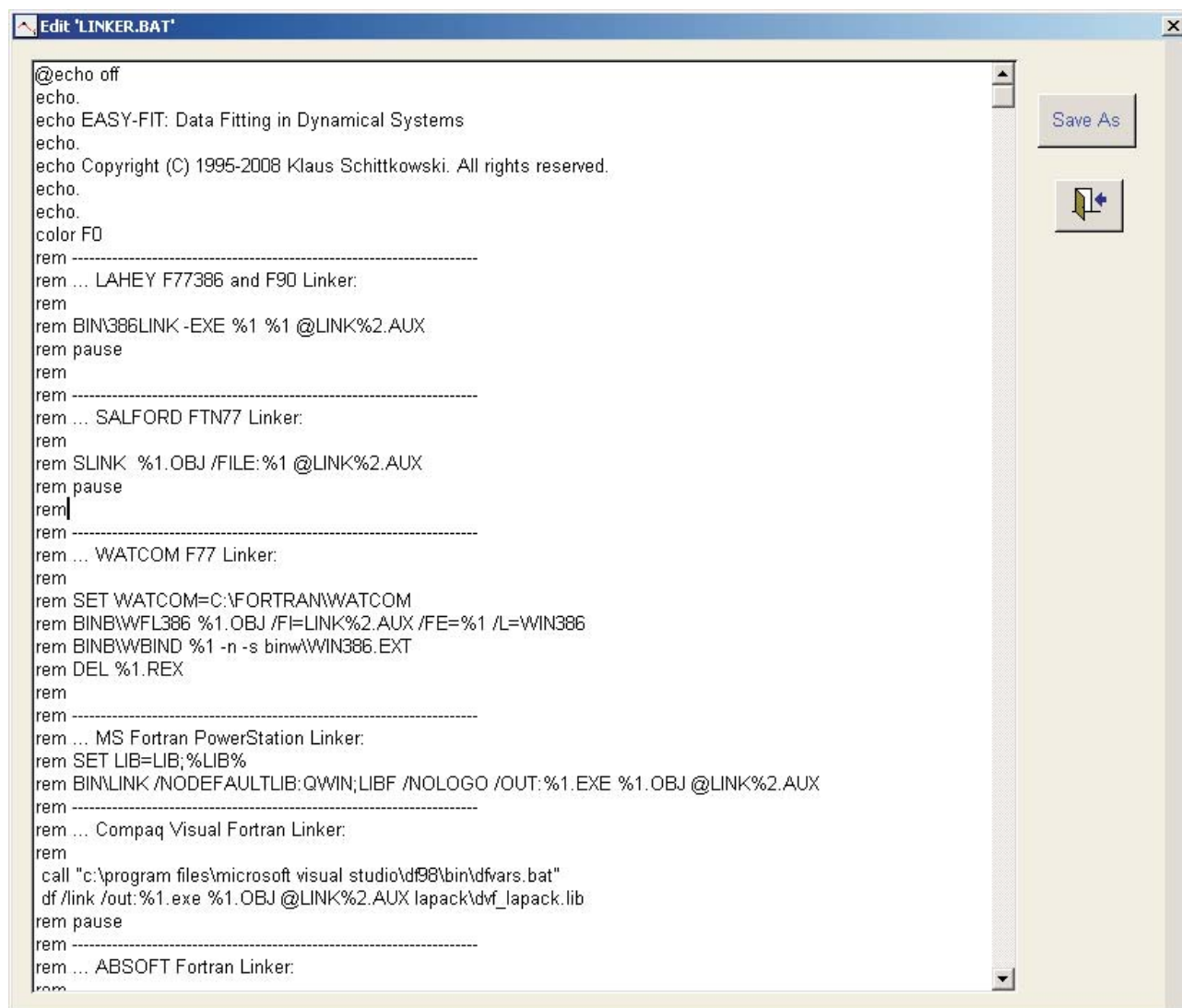
r_i is a uniformly distributed random number between 0 and 1.

b) Normal distribution: The user has to provide desired percentage of standard deviation (variance) t from which a normally distributed error is computed in the form

$$y_i(1 + tr_i/100)$$

where r_i is generated by the uniform normal distribution with mean 0 and variance 1.

Since the original data are lost, it is recommended to save the corresponding columns in the measurement table by the usual *drag and drop* action with respect to any other free column.



```
@echo off
echo.
echo EASY-FIT: Data Fitting in Dynamical Systems
echo.
echo Copyright (C) 1995-2008 Klaus Schittkowski. All rights reserved.
echo.
echo.
color F0
rem -----
rem ... LAHEY F77386 and F90 Linker:
rem
rem BIN386LINK -EXE %1 %1 @LINK%2.AUX
rem pause
rem
rem -----
rem ... SALFORD FTN77 Linker:
rem
rem SLINK %1.OBJ /FILE:%1 @LINK%2.AUX
rem pause
rem
rem -----
rem ... WATCOM F77 Linker:
rem
rem SET WATCOM=C:\FORTRAN\WATCOM
rem BINB\WFL386 %1.OBJ /FI=LINK%2.AUX /FE=%1 /L=WIN386
rem BINB\WBIND %1 -n -s binw\WIN386.EXT
rem DEL %1.REX
rem
rem -----
rem ... MS Fortran PowerStation Linker:
rem SET LIB=LIB;%LIB%
rem BINLINK /NODEFAULTLIB:QWIN;LIBF /NOLOGO /OUT:%1.EXE %1.OBJ @LINK%2.AUX
rem
rem -----
rem ... Compaq Visual Fortran Linker:
rem
rem call "c:\program files\microsoft visual studio\df\bin\dfvars.bat"
rem df /link /out:%1.exe %1.OBJ @LINK%2.AUX lapack\df_lapack.lib
rem pause
rem
rem -----
rem ... ABSOFT Fortran Linker:
rem
```

Figure 8.15: Linker Options


```

C   LWA1   :   LENGTH OF FIRST REAL WORKING ARRAY FOR
C             OPTIMIZATION ROUTINES. MUST BE SUFFICIENTLY BIG.
C   LWA2   :   LENGTH OF SECOND REAL WORKING ARRAY FOR
C             OPTIMIZATION ROUTINES. MUST BE SUFFICIENTLY BIG.
C   LWA3   :   LENGTH OF THIRD REAL WORKING ARRAY USED ONLY FOR
C             EXECUTING ODE-SOLVERS. SIZE DEPENDS ON ODE-METHOD,
C             BANDWIDTH AND DISCRETIZATION ACCURACY.
C   LKWA   :   LENGTH OF INTEGER WORKING ARRAY FOR OPTIMIZATION
C             ROUTINES. MUST BE SUFFICIENTLY BIG.
C   LWLOG  :   LENGTH OF LOGICAL WORKING ARRAY FOR OPTIMIZATION
C             ROUTINES. MUST BE SUFFICIENTLY BIG.
C   LFLUXP :   LENGTH OF WORKING ARRAY FLUXPD FOR STORING
C             DERIVATIVES OF FLUX FUNCTIONS. MUST BE NOT
C             SMALLER THAN 3*NPDE**2*NDIS+NPDE*NDIS.
C
C   IF THE SIZE OF THE LAST 4 PARAMETERS IS TOO SMALL, PDEFIT
C   WILL REPORT AN ERROR MESSAGE.
C
C   IMPORTANT: DO NEVER CHANGE THE COMMON STATEMENTS OR THE
C             CONTENT OF COMMON VARIABLES UNLESS YOU KNOW
C             PRECISELY WHERE THE VARIABLES ARE USED!
C
C   IMPLICIT NONE
C   INTEGER MAXCPL,MAXPAR,MAXPDE,MAXODE,MAXMEA,MAXDIS,MAXTIM,
C   /       MAXOBS,MAXRES,MAXCPB,MAXPOI,MAXPLT,MAXDDP,MAXBPS,
C   /       LWA1,LWA2,LWA3,LKWA,LWLOG,DEMO,MAXDAE,LFLUXP,LMAXIS
C
C   DIMENSIONS:
C
C   PARAMETER(MAXCPL=100,MAXPAR=51,   MAXPDE=50,   MAXODE=2000,
C   /       MAXMEA=30, MAXDIS=1001, MAXTIM=500,   MAXOBS=2000,
C   /       MAXRES=200,MAXCPB=50,   MAXPOI=10,   MAXPLT=200,
C   /       MAXBPS=100,LWA1=3500000,LWA2=1000000,LWA3=10000000,
C   /       LKWA=50000,LWLOG=100000,MAXDDP=1000000,MAXDAE=500,
C   /       LFLUXP=100000, LMAXIS=500)
C
C   LOGICAL PLTEVL
C   INTEGER NMEA, NODE, NTIME, NBAND, NOBS, NPAR, NPAR0, NRES, NEQU, NPLOT,
C   /       NCPL, NCPLO, NCPLA, NPDE, NDIS, NCPB, NCPB2, NBOUND, ITIME,
C   /       INUGRA, IPRINT, NOFUNC, NOGRAD, NOMODF, NOMODG, IERR, ICOUNT,
C   /       DQUMET(MAXPDE, 2, MAXCPB/2), DQUPOI, OPTMET, IOPTP1,
C   /       IOPTP2, IOPTP3, IODEP1, IODEP2, IODEP3, INTEG,
C   /       ICOMP(MAXCPB/2), LBOUND(MAXPDE, MAXCPB/2),
C   /       RBOUND(MAXPDE, MAXCPB/2), IBND(MAXCPB), IXBND(MAXCPB).

```

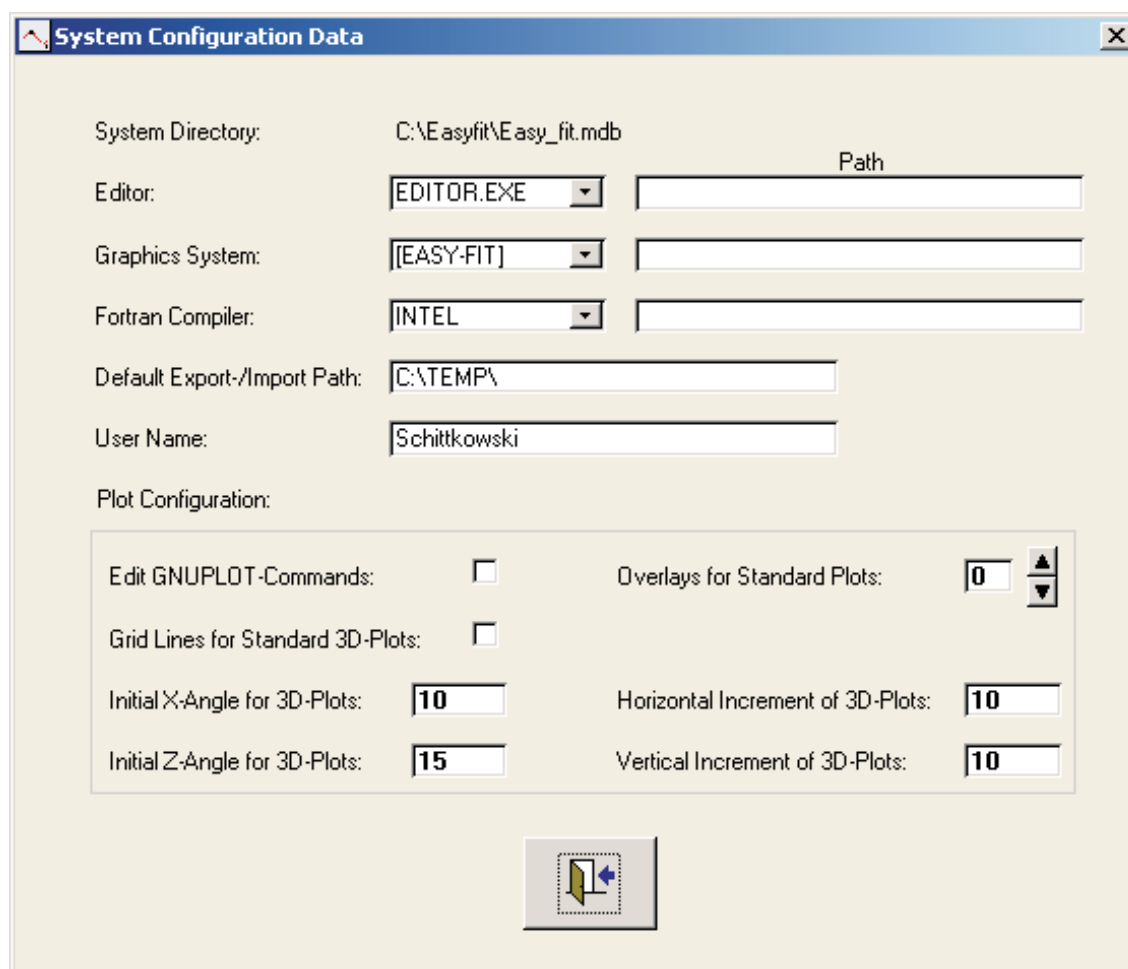
Save As

Compile

Link

↩

Figure 8.16: Dimensioning Parameters



System Configuration Data

System Directory: C:\Easyfit\Easy_fit.mdb

Editor: EDITOR.EXE Path



Graphics System: [EASY-FIT]

Fortran Compiler: INTEL

Default Export-/Import Path: C:\TEMP\

User Name: Schittkowski

Plot Configuration:

Edit GNUPLOT-Commands:	<input type="checkbox"/>	Overlays for Standard Plots:	0	 
Grid Lines for Standard 3D-Plots:	<input type="checkbox"/>			
Initial X-Angle for 3D-Plots:	10	Horizontal Increment of 3D-Plots:	10	
Initial Z-Angle for 3D-Plots:	15	Vertical Increment of 3D-Plots:	10	




Figure 8.17: Configuration Form

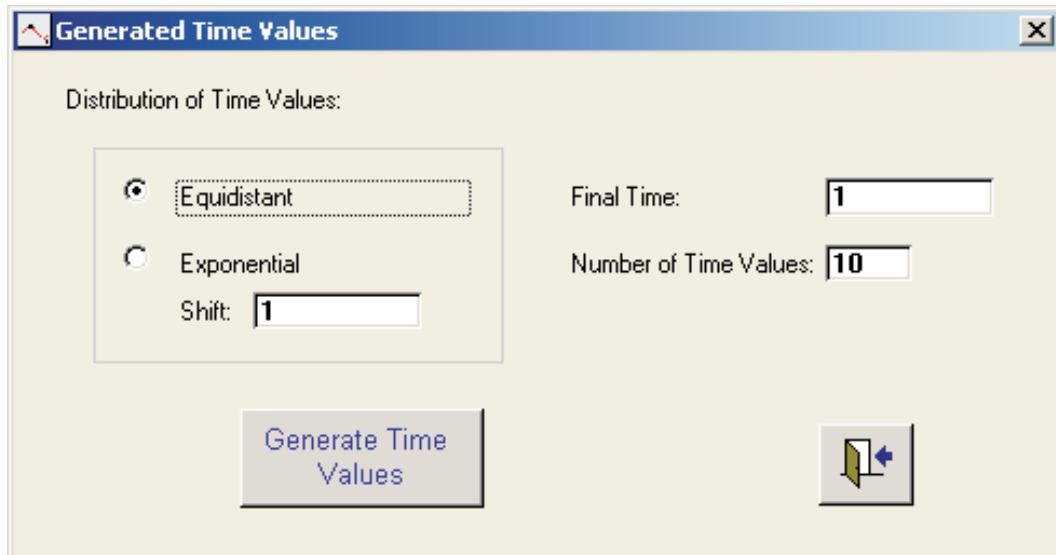


Figure 8.18: Generation of Time Values

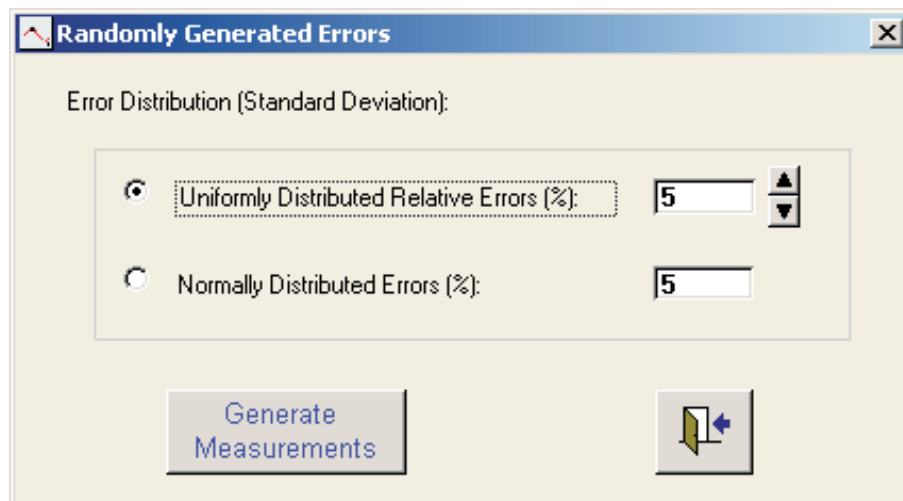


Figure 8.19: Random Errors

Chapter 9

External Usage of Numerical Codes

It is possible to execute the numerical codes MODFIT and PDEFIT also outside of the interactive user interface of **EASY-FIT**^{*ModelDesign*}. A reason could be to solve of a large number of parameter estimation problems controlled by a separate code of the user. In this case, a data input file is requested that contains all information to start the numerical algorithm. The format of the file is documented in this chapter in detail. Test cases illustrate the usage of the codes and their numerical performance.

9.1 MODFIT: Parameter Estimation in Explicit Model Functions, Steady State Systems, Ordinary Differential Equations and Differential Algebraic Systems

Basically MODFIT is a double precision Fortran subroutine and fully documented by initial comments. Since most applications will probably execute the corresponding main program, we describe only the format of the input data and the usage of the subroutine required for model evaluation. More technical information can be retrieved from the initial comments of the main program, for example about linking, parameters, common blocks etc.

An input file named MODFIT.DAT must contain the parameter estimation data, some problem information and optimization data in formatted form. The first 6 columns may contain an arbitrary string to identify the corresponding input row, if allowed by the format.

Line	Format	Name	Description
1	a80	FILE	Name of output files generated by MODFIT. The string may begin with a path name, but must not contain an extension. Suitable extensions are selected by MODFIT.
2	a6,4x,i5	MODEL	Problem name passed to subroutine SYSFUN for identifying data fitting models. The subsequent integer defines the general model structure: 1 - explicit model function 2 - Laplace formulation of model function 3 - steady-state system of equations 4 - system of ordinary differential equations 5 - system of differential algebraic equations
3	a70	INFO	Long information string for plot output.
4	a20	PROJECT	Plot output (first line of information block, e.g., project number).
5	a20	TEST	Plot output (second line of information block, e.g., measurement characterization).
6	a20	DATE	Plot output (third line of information block, e.g., date).
7	a20	USER	Plot output (fourth line of information block, e.g., user name).
8	a10	C-AXIS	Name for c -axis (concentration).
9	a10	T-AXIS	Name for t -axis (time).
10	a6,4x,2i5	NPAR	Number of parameters to be optimized, must be at least one.
		NBPV	Number of variable break points, i.e., the last NBPV variables are treated as break points where integration is restarted.
11	a6,4x,i5	NRES	Total number of constraints without bounds.

Line	Format	Name	Description
12	a6,4x,i5	NEQU	Number of equality constraints.
13	a6,4x,2g20.4	RT,RC	Formatted input of NRES rows each containing two real numbers identifying the experimental time and concentration parameters for which a constraint is to be supplied. The order is arbitrary, but first the equality and subsequently the inequality constraints are to be defined. The data are rounded to the nearest actual time and concentration value.
14	a6,4x,i5	NODE	Number of differential equations. NODE can be zero if no differential equation is defined.
15	a6,4x,i5	NCONC	Number of concentration values. NCONC must be -1 or higher.
16	a6,4x,2i5	NTIME, MPLOT	Number of time points, must be greater than 0. Logarithmic scaling of x-axis (MPLOT=1) or not (MPLOT=0).
17	a6,4x,i5	NMEAS	Number of measurement sets, i.e., of model functions with respect to which measurements are supplied. NMEAS is the dimension of the fitting function.
18	a6,4x,i5	NPLOT	Number of plot points to be computed by additional model function evaluations. Plots are generated by interpolation (linear or polynomial depending on graphics system). NPLOT may be zero if no plot data are required.
19	a6,4x,i5	NOUT	Output flag for MODFIT: NOUT=0 - No output generated on files '*.PRT', '*.TEX'. and '*.RPL' NOUT=1 - Output generated on files '*.PRT', '*.TEX'. and '*.RPL'

Line	Format	Name	Description
20	a6,4x,4i5	METHOD, ISHT, NORM, NUMGRA	<p>Choice of simulation or optimization algorithm:</p> <p>METHOD=0 - Simulation</p> <p>METHOD=1 - Call of DFNLP (Schittkowski [429])</p> <p>METHOD=2 - dummy</p> <p>ISHT is the shooting index to identify number and position of shooting points. If ISHT>0, only METHOD=0, 1, or 2 are allowed.</p> <p>ISHT=0 - no shooting at all</p> <p>ISHT=1 - shooting at every measurement time</p> <p>ISHT=2 - shooting at every second time</p> <p>ISHT=3 - shooting at every third time</p> <p>etc.</p> <p>NORM determines the data fitting norm.</p> <p>NORM=1 = L_1 norm, sum of absolute residuals</p> <p>NORM=2 = L_2 norm, sum of squared residuals</p> <p>NORM=3 = L_∞ norm, maximum of absolute residuals</p> <p>NUMGRA must be set for gradient evaluation.</p> <p>NUMGRA=-1 - analytical derivatives available</p> <p>NUMGRA=0/1 - forward differences</p> <p>NUMGRA=2 - two-sided differences</p> <p>NUMGRA=3 - 5-point difference formula</p>
21	a6,4x,i5	OPTP1	<p>Parameter for chosen optimization algorithm:</p> <p>METHOD=0 - significance level (0/1/5/10)</p>

Line	Format	Name	Description
			METHOD=1 - maximum number of iterations
22	a6,4x,i5	OPTP2	Parameter for chosen optimization algorithm: METHOD=1 - maximum number of line search steps
23	a6,4x,i5	OPTP3	Output level for chosen optimization algorithm, usage defined in documentation of optimization algorithm executed. The output is directed to a file with extension HIS. Only the residuals are displayed on screen.
24	a6,4x,g10.4	OPTE1	Tolerance for chosen optimization algorithm: METHOD=0 - tolerance for rank determination METHOD=1 - final termination tolerance
25	a6,4x,g10.4	OPTE2	Tolerance for chosen optimization algorithm: METHOD=1 - expected size of residual
26	a6,4x,g10.4	OPTE3	Tolerance for chosen optimization algorithm: METHOD=1 - internal scaling bound
27	a6,4x,i5	ODEP1	Parameter for selection of differential equation solver: ODEP1=1 - dummy ODEP1=2 - dummy ODEP1=3 - dummy ODEP1=4 - RADAU5 (implicit Runge-Kutta, order 5) ODEP1=5 - dummy ODEP1=6 - dummy ODEP1=11- IND-DIR (Runge-Kutta 5-th order with internal sensitivity analysis)
28	a6,4x,5i5	ODEP2	Order of ODE-method or gradient evaluation, respectively:

Line	Format	Name	Description
		NDAE	0 - no derivatives 1 - derivatives of right hand side supplied Number of algebraic equations (in case of DAE)
		IND1	Number of index-1-variables (in case of DAE)
		IND2	Number of index-2-variables (in case of DAE)
		IND3	Number of index-3-variables (in case of DAE)
			If NDAE=NODE, it is assumed that a steady state system is to be solved.
29	a6,4x,i5	ODEP3	Approximate number of correct digits when gradients must be evaluated numerically by forward differences. If set to zero, a suitable tolerance is internally computed
30	a6,4x,5i5	ODEP4	Bandwidth of Jacobian of right-hand side (only for implicit solver), must be smaller than NODE. In case of ODEP4=0, usage of full matrix is assumed.
31	a6,4x,g10.4	ODEE1	Final termination accuracy for ODE-solver with respect to the relative global error.
32	a6,4x,g10.4	ODEE2	Final termination accuracy for ODE-solver with respect to the absolute global error.
33	a6,4x,g10.4	ODEE3	Tolerance for solving differential equation: initial step-size
34	a6,4x,3g20.8	XL,X,XU	Formatted input of NPAR rows each containing three real numbers for - lower bound for estimated parameter - starting value for estimated parameter - upper bound for estimated parameter If the file '*.RES' contains the results of a previous run, then the corresponding parameter values are read and replace the given ones, i.e., the X-values.

Line	Format	Name	Description
35	a6,4x,i5	SCALE	<p>Scale for weight factors:</p> <p>0 - no additional scaling</p> <p>1 - division by square root of sum of squared measurements</p> <p>-1 - division by absolute measurement value</p> <p>-2 - division by squared measurement value</p>
36	*		<p>In case of NCONC>0, unformatted input of NTIME*NCONC rows for $j = 1$ to NCONC and $i = 1$ to NTIME (in this order) with the following data:</p> <p>t_i - i-th measurement time, not smaller than zero</p> <p>c_j - j-th concentration value</p> <p>y_{ij}^k, w_{ij}^k - measured data, i.e., experimental output, and individual weight factor for measurement with number $k, k = 1, \dots, \text{NMEAS}$</p> <p>Otherwise these lines contain the following data in unformatted form for $i = 1$ to NTIME :</p> <p>t_i - i-th measurement time, not smaller than zero</p> <p>y_i^k, w_i^k - measured data, i.e., experimental output, and individual weight factor for measurement with number $k, k = 1, \dots, \text{NMEAS}$</p> <p>Note that the time values must increase. In case of NCONC > 0, the concentration values must increase as well and the set of time values must be repeated for each concentration.</p>
37	a6,4x,i5	NLPIP	<p>Output flag for NLPQLP when computing consistent initial values in case of a DAE or solving system of nonlinear equations in case of steady state system:</p>

Line	Format	Name	Description
			NLPIP=0 - no output at all NLPIP=1 - only final summary NLPIP=2 - one line per iteration NLPIP=3 - detailed output per iteration
38	a6,4x,i5	NLPMI	Maximum number of iterations for NLPQLP when computing consistent initial values in case of DAE.
39	a6,4x,g10.4	NLPAC	Final termination accuracy for for NLPQLP when computing consistent initial values in case of DAE.
40	a6,4x,i5	NBPC	Number of constant break points where integration is restarted with initial tolerances. Constant break points are permitted only if NBPV=0 and ODEP1<7.
41	*		Unformatted input of NBPC rows each containing one time value in increasing order that represents a break point of the right-hand side of an ODE/DAE.

Among the data generated by MODFIT, are result, report and plot files, that can be used for external programs evaluating these data. The format is described in detail among the initial comments of the Fortran code of MODFIT. Numerical results are stored on a file with extension .RES, and are read by **EASY-FIT**^{ModelDesign} as soon as numerical execution is terminated. In case of a simulation run with a positive significance level, statistical data are written to a file with extension .STA in abbreviated form.

The code distributed together with **EASY-FIT**^{ModelDesign}, allows the input of model functions on a user provided file with name <MODEL>.FUN in a directory specified in the first line of MODFIT.DAT. In this case, the input format is the PCOMP language must be used as outlined in the previous chapters. A specific advantage is the automatic evaluation of derivatives. There is no further compilation or link necessary and MODFIT can be started immediately, as soon as both input files are created.

On the other hand a user has the option to implement all model functions in form of Fortran code, and to link his own module to the object file of MODFIT. Information about the remaining files to be linked in this case, is found among the initial comments of the file MODFIT.FOR that contains the main program. The model data, i.e., fitting criterion, system equations, bounds and initial values, must be provided by a user-defined subroutine called SYSFUN:

```
SUBROUTINE  SYSFUN (NP,MAXP,NO,MAXO,NF,MAXF,NR,MAXR,X,Y,
                    T,C,YP,Y0,FIT,G,DYP,DY0,DFIT,DG,IFLAG)
```

The meaning of the arguments is as follows:

NP	: Number of parameters in array X.
MAXP	: Dimensioning parameter, must be greater or equal to NP.
NO	: Number of equations of dynamical system.
MAXO	: Dimensioning parameter, must be greater or equal to NO.
NF	: Number of fitting functions in the parameter estimation problem that corresponds to the number of measurement sets.
MAXF	: Dimensioning parameter, must be greater or equal to NF.
NR	: Number of constraints of the parameter estimation problem.
MAXR	: Dimensioning parameter, must be greater or equal to NR.
X(MAXP)	: When calling SYSFUN, X contains the NP coefficients of the actual variables to be estimated. X is not allowed to be altered within the subroutine.
Y(MAXO)	: When calling SYSFUN, Y contains the NO coefficients of the differential equation on the right-hand side. Y is not allowed to be altered within the subroutine.
T	: <i>Time</i> variable.
C	: <i>Concentration</i> variable.
YP(MAXO)	: Function values to be evaluated in case of 'IFLAG=1' for right-hand side of a dynamical system.
Y0(MAXO)	: Function values to be evaluated in case of 'IFLAG=2' for initial values of the dynamical system.
FIT(MAXF)	: Function values to be evaluated in case of 'IFLAG=3' for the NFIT fitting conditions in a parameter estimation problem.
G(MAXR)	: Function values to be evaluated in case of 'IFLAG=4' for constraints in the parameter estimation problem.
DYP(MAXO,M1)	: Gradient values to be evaluated in case of 'IFLAG=5' for the right-hand side of the dynamical system for variables X and Y, and for Y only in case of 'IFLAG=9', where M1=MAXP+NO.
DY0(MAXO,NP)	: Gradient values to be evaluated in case of 'IFLAG=6' for initial values with respect to variable X.
DFIT(MAXF,M1)	: Gradient values to be evaluated in case of 'IFLAG=7' for the fitting criteria of the parameter estimation problem, with respect to X and Y, where M1=MAXP+NO.

- DG(MAXR,NP) : Gradient values to be evaluated in case of 'IFLAG=8' for the constraints of the parameter estimation problem, with respect to X.
- IFLAG : Flag defining the desired type of calculation.
- IFLAG=0: Execution of SYSFUN before requiring function or gradient values, e.g., for preparing common's.
- IFLAG=1: Evaluate right-hand side of dynamical system and store results in YP.
- IFLAG=2: Evaluate initial values of dynamical system and store results in Y0.
- IFLAG=3: Evaluate fitting criteria and store results in FIT.
- IFLAG=4: Evaluate constraints and store results in G.
- IFLAG=5: Evaluate gradients of the right-hand side with respect to X and Y, and store results in DYP.
- IFLAG=6: Evaluate gradients of initial values with respect to X, and store results in DY0.
- IFLAG=7: Evaluate gradients of fitting criteria with respect to X and Y, and store results in DFIT.
- IFLAG=8: Evaluate gradients of constraints with respect to X, and store results in DG.
- IFLAG=9: Evaluate gradients of the right-hand side with respect to Y only, and store results in DYP.

Subroutine SYSFUN is called with IFLAG=5 only if an ODE-solver with internal numerical differentiation is used, i.e., IND-DIR. On the other hand, derivatives are needed only if an implicit ODE-solver is executed in case of IFLAG=9.

There are some files with names MODFUN_E.FOR and MODFUN_O.FOR distributed together with **EASY-FIT** *ModelDesign* that contain source codes for very simple examples to illustrate the usage of user-provided Fortran code. The input of data and model functions is to be outlined in addition by some examples.

Example 9.1 (DFE1) *The first one consists of an explicit model function given in the form*

$$h(p, t) = a(\exp(-\beta t) + 0.1) \exp(-a_1 t) + b \exp(-b_1 t) + c \exp(-c_1 t) \sin(dt) . \quad (9.1)$$

The input file must contain then the following information:

```
C:\EASYFIT\PROBLEMS\DFE1
DFE1      1
Test Problem DFE1
Demo
Artificial Data
```

10.2.1994

Schittkowski

x

t

NPAR	8	0			
NRES	0				
NEQU	0				
NODE	0				
NCONC	0				
NTIME	10	0			
NMEAS	1				
NPLOT	50				
NOUT	1				
METHOD	1				
OPTP1	100				
OPTP2	8				
OPTP3	2				
OPTE1	1.0E-10				
OPTE2	1.0				
OPTE3	0.0				
ODEP1	0	0	0	0	0
ODEP2	0				
ODEP3	0				
ODEP4	0				
ODEE1	0.0				
ODEE2	0.0				
ODEE3	0.0				
a	0.0		1.0		1000.0
bt	1.0E-5		0.1		1000.0
a1	0.0001		0.001		1000.0
b	0.001		5.0		10000.0
b1	1.0E-5		100.0		10000.0
c	0.0		0.0		1000.0
c1	1.0E-5		0.1		1000.0
d	0.0		0.00001		10000.0
SCALE	1				
0.0	25.00	1.0			
3.0	15.51	1.0			
6.0	16.08	1.0			
9.0	16.11	1.0			
12.0	15.48	1.0			
15.0	14.24	1.0			
18.0	12.50	1.0			
21.0	10.41	1.0			
22.5	9.299	1.0			
24.0	8.162	1.0			
NLPIP =	0				
NLPMI =	0				
NLPAC =	0.0				

The code DFNLP computed a solution in 24 iterations, and the optimal fit is shown in Figure 9.1. The corresponding Fortran subroutine SYSFUN can be implemented as follows:

```

      SUBROUTINE SYSFUN(NP,MAXP,NO,MAXO,NF,MAXF,NR,MAXR,X,Y,T,
/
      CONC,YP,YO,FIT,G,DYP,DYO,DFIT,DG,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(MAXP),Y(MAXO),YO(MAXO),YP(MAXO),G(MAXR),
/
      FIT(MAXF),DYO(MAXO,MAXP),
/
      DYP(MAXO,MAXP+MAXO),DG(MAXR,MAXP),
/
      DFIT(MAXF,MAXP+MAXO)
C
C   BRANCH W.R.T. IFLAG
C
      IF (IFLAG.EQ.0) RETURN
C
      A = X(1)
      BT = X(2)
      A1 = X(3)
      B = X(4)
      B1 = X(5)
      C = X(6)
      C1 = X(7)
      D = X(8)
C
      GOTO (100,200,300,400,500,600,700,800,900), IFLAG
C
C   RIGHT-HAND SIDE OF ODE
C
100 CONTINUE
      RETURN
C
C   INITIAL VALUES FOR ODE
200 CONTINUE
      RETURN
C
C   FITTING CRITERIA
C
300 CONTINUE
      FIT(1) = A*(DEXP(-BT*T) + 0.1)*DEXP(-A1*T) + B*DEXP(-B1*T)
/
      + C*DEXP(-C1*T)*DSIN(D*T)
      RETURN
C
C   CONSTRAINTS
C
400 CONTINUE
      RETURN
C

```

```

C   GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. X AND Y
C
500 CONTINUE
    RETURN
C
C   GRADIENTS OF INITIAL VALUES OF ODE W.R.T. X
C
600 CONTINUE
    RETURN
C
C   GRADIENTS OF FITTING CRITERIA W.R.T. X
C
700 CONTINUE
    DFIT(1,1) = (DEXP(-BT*T) + 0.1)*DEXP(-A1*T)
    DFIT(1,2) = -T*A*DEXP(-BT*T)*DEXP(-A1*T)
    DFIT(1,3) = -T*A*(DEXP(-BT*T) + 0.1)*DEXP(-A1*T)
    DFIT(1,4) = DEXP(-B1*T)
    DFIT(1,5) = -T*B*DEXP(-B1*T)
    DFIT(1,6) = DEXP(-C1*T)*DSIN(D*T)
    DFIT(1,7) = -T*C*DEXP(-C1*T)*DSIN(D*T)
    DFIT(1,8) = D*C*DEXP(-C1*T)*DCOS(D*T)
    RETURN
C
C   GRADIENTS OF CONSTRAINTS
C
800 CONTINUE
    RETURN
C
C   GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. Y
C
900 CONTINUE
    RETURN
END

```

Example 9.2 (SE) *The second example describes a single differential equation with three concentration values,*

$$\dot{y} = k_1(r - y)(c - y) - k_2y \quad . \quad (9.2)$$

Here the coefficients k_1 , k_2 , and r are to be estimated. The input file is the following one:

```

C:\EASYFIT\PROBLEMS\SE
SE          4
Test Problem SE
Demo
Artificial Data
10.2.1994
Schittkowski
c

```


t					
NPAR	3	0			
NRES	0				
NEQU	0				
NODE	1				
NCONC	3				
NTIME	14	0			
NMEAS	1				
NPLOT	50				
NOUT	1				
METHOD	1				
OPTP1	100				
OPTP2	8				
OPTP3	2				
OPTE1	1.0E-9				
OPTE2	1.0				
OPTE3	0.0				
ODEP1	11				
ODEP2	1	0	0	0	0
ODEP3	6				
ODEP4	0				
ODEE1	1.0E-9				
ODEE2	1.0E-9				
ODEE3	1.0E-6				
Kp	1.0E-4		1.0E-4		10.0
Km	1.0E-6		1.0E-5		0.01
R0	100.0		150.0		400.0
SCALE	1				
1.0	100.0	1.39	1.0		
3.0	100.0	4.06	1.0		
5.0	100.0	5.24	1.0		
10.0	100.0	11.6	1.0		
15.0	100.0	11.2	1.0		
20.0	100.0	17.4	1.0		
25.0	100.0	0.0	0.0		
30.0	100.0	20.6	1.0		
35.0	100.0	25.1	1.0		
40.0	100.0	25.3	1.0		
50.0	100.0	26.9	1.0		
60.0	100.0	0.0	0.0		
90.0	100.0	34.1	1.0		
105.0	100.0	0.0	0.0		
1.0	1000.0	16.5	1.0		
3.0	1000.0	39.1	1.0		
5.0	1000.0	57.4	1.0		
10.0	1000.0	97.3	1.0		
15.0	1000.0	118.5	1.0		
20.0	1000.0	139.6	1.0		
25.0	1000.0	159.8	1.0		

30.0	1000.0	173.2	1.0
35.0	1000.0	180.3	1.0
40.0	1000.0	189.9	1.0
50.0	1000.0	200.4	1.0
60.0	1000.0	209.3	1.0
90.0	1000.0	202.4	1.0
105.0	1000.0	213.1	1.0
1.0	5000.0	72.9	1.0
3.0	5000.0	153.8	1.0
5.0	5000.0	189.4	1.0
10.0	5000.0	239.7	1.0
15.0	5000.0	250.6	1.0
20.0	5000.0	234.1	1.0
25.0	5000.0	255.1	1.0
30.0	5000.0	254.7	1.0
35.0	5000.0	266.2	1.0
40.0	5000.0	270.3	1.0
50.0	5000.0	0.0	0.0
60.0	5000.0	0.0	0.0
90.0	5000.0	237.7	1.0
105.0	5000.0	250.8	1.0
NLPIP =	0		
NLPMI =	0		
NLPAC =	0.0		
NBPC	0		

A solution is obtained by DFNLP in 19 iterations, see Figure 9.2. The Fortran code for the function and gradient evaluation is defined as follows:

```

      SUBROUTINE SYSFUN(NP,MAXP,NO,MAXO,NF,MAXF,NR,MAXR,
/
      X,Y,T,C,YP,YO,FIT,G,DYP,DYO,DFIT,DG,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION K1,K2
      DIMENSION X(MAXP),Y(MAXO),YO(MAXO),YP(MAXO),G(MAXR),
/
      FIT(MAXF),DYO(MAXO,MAXP),
/
      DYP(MAXO,MAXP+MAXO),DG(MAXR,MAXP),
/
      DFIT(MAXF,MAXP+MAXO)
C
C   BRANCH W.R.T. IFLAG
C
      IF (IFLAG.EQ.0) RETURN
      K1 = X(1)
      K2 = X(2)
      R  = X(3)
      GOTO (100,200,300,400,500,600,700,800,900), IFLAG
C
C   RIGHT-HAND SIDE OF ODE
C
100 CONTINUE

```

```

        YP(1) = K1*(R - Y(1))*(C - Y(1)) - K2*Y(1)
        RETURN
C
C   INITIAL VALUES FOR ODE
C
200 CONTINUE
    YO(1) = 0.0
    RETURN
C
C   FITTING CRITERIA
C
300 CONTINUE
    FIT(1) = Y(1)
    RETURN
C
C   CONSTRAINTS
C
400 CONTINUE
    RETURN
C
C   GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. X AND Y
C
500 CONTINUE
    DYP(1,1) = (R - Y(1))*(C - Y(1))
    DYP(1,2) = -Y(1)
    DYP(1,3) = K1*(C - Y(1))
    DYP(1,4) = -K1*(C - Y(1)) - K1*(R - Y(1)) - K2
    RETURN
C
C   GRADIENTS OF INITIAL VALUES OF ODE W.R.T. X
C
600 CONTINUE
    DYO(1,2) = 0.0
    DYO(1,3) = 0.0
    RETURN
C
C   GRADIENTS OF FITTING CRITERIA W.R.T. X
C
700 CONTINUE
    DFIT(1,1) = 0.0
    DFIT(1,2) = 0.0
    DFIT(1,3) = 0.0
    DFIT(1,4) = 1.0
    RETURN
C
C   GRADIENTS OF CONSTRAINTS
C
800 CONTINUE
    RETURN

```

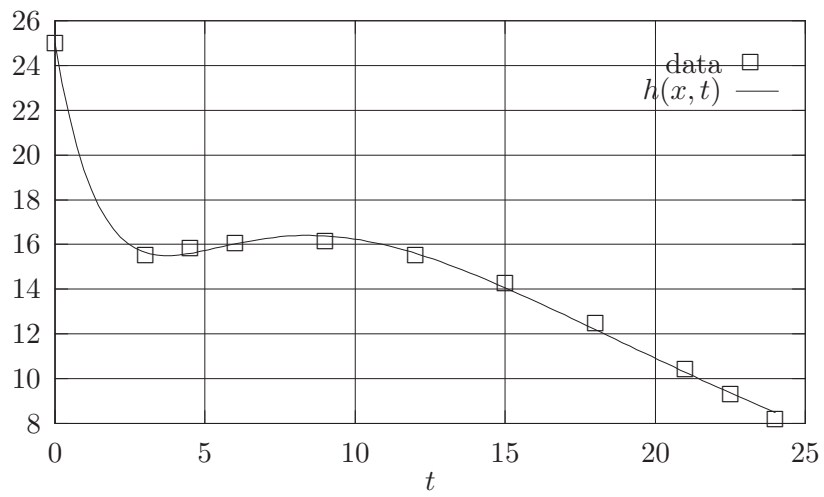


Figure 9.1: Final Trajectory for Problem DFE1

```

C
C  GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. Y
C
900 CONTINUE
    DYP(1,1) = -K1*(C - Y(1)) - K1*(R - Y(1)) - K2
    RETURN
C
    END

```

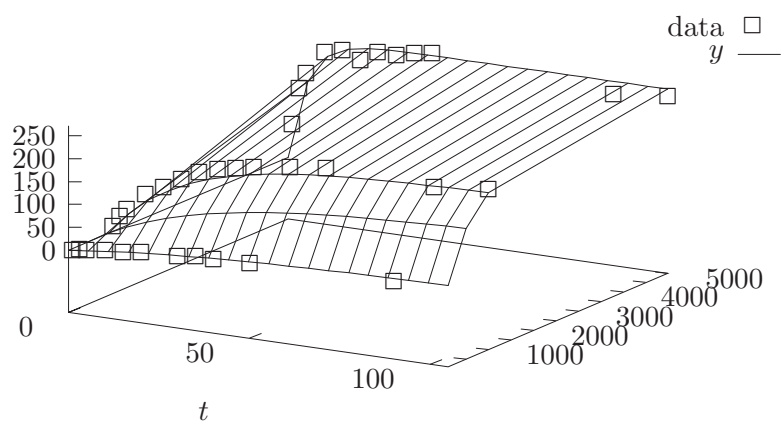


Figure 9.2: Final Trajectory for Problem SE

Example 9.3 (VDPOL) *The input of data and model functions in case of a differential algebraic system, is to be outlined by an example, the van der Pol's equation, see also Section 5.5. The model function is*

$$\dot{y} = z \quad , \quad \dot{z} = y - a(1 - y^2)z \quad . \quad (9.3)$$

We choose the consistent initial values

$$y^0 = a \quad , \quad z^0 = b/(a(1 - b^2))$$

and consider a and b as parameters to be estimated. The fitting criteria are the solutions y and z . The data input file MODFIT.DAT has the following structure:

```

C:\EASYFIT\PROBLEMS\VDPOL
VDPOL          5
van der Pol's equation, electrical circuit
Demo
Schittkowski
Simulation
-
-
t
NPAR           2      0
NRES           0
NEQU           0
NODE           2
NCONC          0
NTIME          5      0
NMEAS          2
NPLOT          50
NOUT           0
METHOD         1
OPTH1          40
OPTH2           8
OPTH3           2
OPTE1          1.0E-07
OPTE2          1.0E-01
OPTE3          1.0E+02
ODEP1           5
ODEP2           1      1      2      0      0
ODEP3           6
ODEP4           0
ODEE1          1.0E-09
ODEE2          1.0E-06
ODEE3          1.0E-04
a              5.0E-01      1.001      10.0
b              1.5        2.001      5.0
SCALE         -1

```

0.0	2.000	1.0	-6.667E-1	1.0
2.0E-1	1.858	1.0	-7.575E-1	1.0
4.0E-1	1.693	1.0	-9.069E-1	1.0
6.0E-1	1.485	1.0	-1.233	1.0
8.0E-1	1.084	1.0	-6.200	1.0
NLPIP	0			
NLPMI	50			
NLPAC	1.0E-10			
NBPC	0			

The code *DFNLP* computed a solution in 4 iterations. The optimal fit is shown in Figure 9.3. The corresponding Fortran subroutine *SYSFUN* needs gradients only for *IFLAG*=9, i.e., for gradients of the right-hand side of the DAE subject to the state variables *y* and *z*, since internal numerical differentiation is not allowed.

```

      SUBROUTINE SYSFUN(NP,MAXP,NO,MAXO,NF,MAXF,NR,MAXR,
/
      X,Y,T,C,YP,YO,FIT,G,DYP,DYO,DFIT,DG,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(MAXP),Y(MAXO),YO(MAXO),YP(MAXO),G(MAXR),
/
      FIT(MAXF),DYO(MAXO,MAXP),
/
      DYP(MAXO,MAXP+MAXO),DG(MAXR,MAXP),
/
      DFIT(MAXF,MAXP+MAXO)
C
C   BRANCH W.R.T. IFLAG
C
      IF (IFLAG.EQ.0) RETURN
      A = X(1)
      B = X(2)
      GOTO (100,200,300,400,500,600,700,800,900), IFLAG
C
C   RIGHT-HAND SIDE OF ODE
C
100 CONTINUE
      YP(1) = Y(2)
      YP(2) = A*(1.0 - Y(1)**2)*Y(2) - Y(1)
      RETURN
C
C   INITIAL VALUES FOR ODE
C
200 CONTINUE
      YO(1) = B
      YO(2) = B/(A*(1-B*B))
      RETURN
C
C   FITTING CRITERIA
C
300 CONTINUE
      FIT(1) = Y(1)

```

```

        FIT(2) = Y(2)
        RETURN
C
C   CONSTRAINTS
C
    400 CONTINUE
        RETURN
C
C   GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. X AND Y
C
    500 CONTINUE
        RETURN
C
C   GRADIENTS OF INITIAL VALUES OF ODE W.R.T. X
C
    600 CONTINUE
        RETURN
C
C   GRADIENTS OF FITTING CRITERIA W.R.T. X AND Y
C
    700 CONTINUE
        RETURN
C
C   GRADIENTS OF CONSTRAINTS
C
    800 CONTINUE
        RETURN
C
C   GRADIENTS OF RIGHT-HAND SIDE OF ODE W.R.T. Y
C
    900 CONTINUE
        DYP(1,1) = 0.0
        DYP(1,2) = 1.0
        DYP(2,1) = -2.0*A*Y(1)*Y(2) - 1.0
        DYP(2,2) = A*(1.0 - Y(1)**2)
        RETURN
C
        END

```

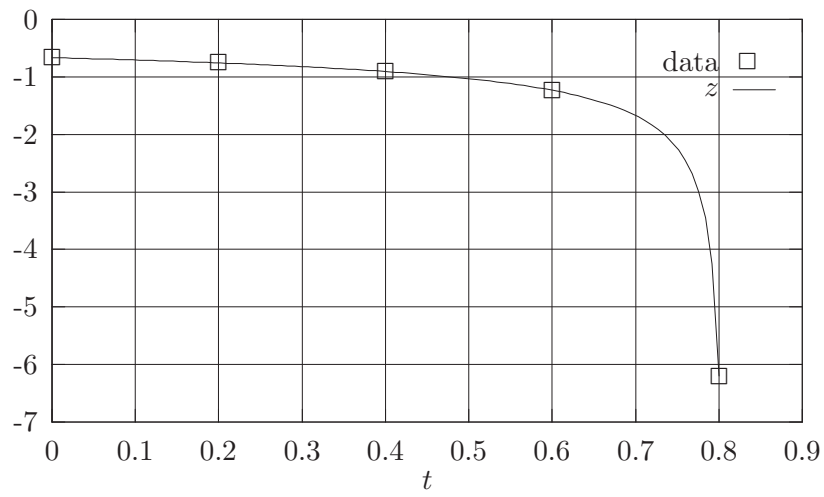
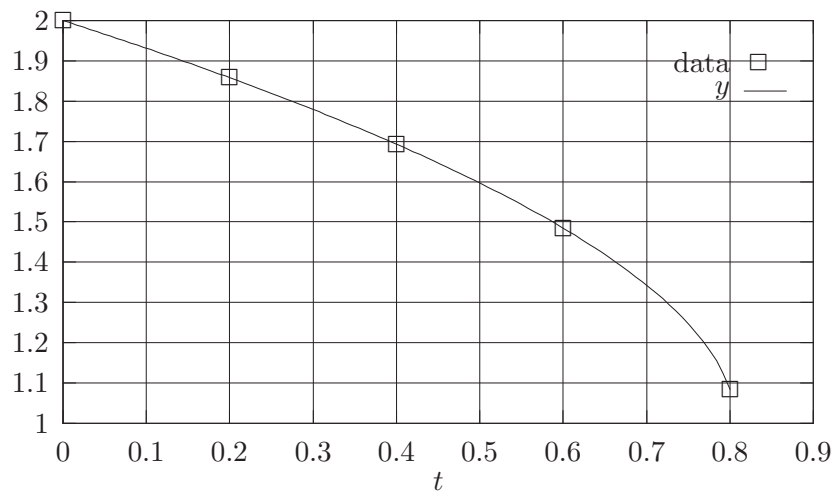



Figure 9.3: Final Trajectories for Problem VDPOL

Example 9.4 (RECLIG10) *To illustrate the implementation of a steady state system, we consider the following example that is similar to a receptor-ligand binding study with one receptor and two ligands,*

$$\begin{aligned} z_1(1 + p_1 z_2 + p_2 z_3) - p_3 &= 0, \\ z_2(1 + p_1 z_1) - p_4 &= 0, \\ z_3(1 + p_2 z_1) - t &= 0. \end{aligned} \tag{9.4}$$

The system parameters are z_1 , z_2 and z_3 , and the parameters to be estimated, are p_1 , p_2 , p_3 , and p_4 . t is the independent model or time variable to be replaced by experimental data. The fitting criterion is $\bar{h}(p, z, t) = p_4 - z_2$ and we use the starting values $z_1^0 = p_3$, $z_2^0 = p_4$ and $z_3^0 = t$ for solving the system of nonlinear equations.

The subsequent input file shows the parameters, tolerances and measurement values used. The data fitting code DFNLP needed 20 iterations to satisfy the stopping conditions subject to the tolerance given. The corresponding data and function plot is found in Figure 9.4.

```
C:\EASYFIT\problems\DYN_EQ
RECLIG10      2
Steady state system, receptor-ligand binding study
Demo
Schittkowski
Simulation
nMol
Null
log nMol
NPAR  =      4      0
NRES  =      0
NEQU  =      0
NODE  =      3
NCONC =      0
NTIME =     13      0
NMEAS =      1
NPLOT =     50      0
NOUT  =      0
METHOD=     01      0
OPTP1 =     200
OPTP2 =      20
OPTP3 =      02
OPTE1 =     1.0E-09
OPTE2 =     1.0E+00
OPTE3 =     1.0E+02
ODEP1 =      0
ODEP2 =      1      3      0      0      0
ODEP3 =      0
ODEP4 =      0
ODEE1 =     0.0
ODEE2 =     0.0
```

```

ODEE3 =    0.0
p1      0.0          1.0          10000.0
p2      0.0          1.0          10000.0
p3      0.0          100.0         10000.0
p4      0.0          2.0          10000.0
SCALE =    1
1.0      0.332    1.0
5.0      0.331    1.0
1.0E+1    0.331    1.0
5.0E+1    0.327    1.0
1.0E+2    0.321    1.0
5.0E+2    0.289    1.0
1.0E+3    0.250    1.0
5.0E+3    0.125    1.0
1.0E+4    0.077    1.0
5.0E+4    0.019    1.0
1.0E+5    0.010    1.0
5.0E+5    0.002    1.0
1.0E+6    0.001    1.0
NLPIP      0
NLPMI     100
NLPAC     1.0E-11
NDISCO=    0

```

The corresponding system functions must be programmed in the following way:

```

      SUBROUTINE SYSFUN(NP,MAXP,NO,MAXO,NF,MAXF,NR,MAXR,
/
/          X,Y,T,C,YP,YO,FIT,G,DYP,DYO,DFIT,DG,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(MAXP),Y(MAXO),YO(MAXO),YP(MAXO),G(MAXR),
/
/          FIT(MAXF),DYO(MAXO,MAXP),
/
/          DYP(MAXO,MAXP+MAXO),DG(MAXR,MAXP+MAXO),
/
/          DFIT(MAXF,MAXP+MAXO)
C
C   BRANCH W.R.T. IFLAG
C
      IF (IFLAG.EQ.0) RETURN
      GOTO (100,200,300,400,500,600,700,800,900), IFLAG
C
C   RIGHT-HAND SIDE OF ODE
C
100 CONTINUE
      YP(1) = Y(1)*(1.0 + X(1)*Y(2) + X(2)*Y(3)) - X(3)
      YP(2) = Y(2)*(1.0 + X(1)*Y(1)) - X(4)
      YP(3) = Y(3)*(1.0 + X(2)*Y(1)) - T
      RETURN
C
C   STARTING VALUES

```

```

C
  200 CONTINUE
      Y0(1) = X(3)
      Y0(2) = X(4)
      Y0(3) = T
      RETURN

C
C   FITTING CRITERIA
C
  300 CONTINUE
      FIT(1) = X(4) - Y(2)
      RETURN

C
C   CONSTRAINTS
C
  400 CONTINUE
      RETURN

C
C   GRADIENTS OF RIGHT-HAND SIDE OF EQUATIONS W.R.T. X AND Y
C
  500 CONTINUE
      DYP(1,1) = Y(1)*Y(2)
      DYP(1,2) = Y(1)*Y(3)
      DYP(1,3) = -1.0
      DYP(1,4) = 0.0
      DYP(1,5) = 1.0 + X(1)*Y(2) + X(2)*Y(3)
      DYP(1,6) = Y(1)*X(1)
      DYP(1,7) = Y(1)*X(2)
      DYP(2,1) = Y(2)*Y(1)
      DYP(2,2) = 0.0
      DYP(2,3) = 0.0
      DYP(2,4) = -1.0
      DYP(2,5) = Y(2)*X(1)
      DYP(2,6) = 1.0 + X(1)*Y(1)
      DYP(2,7) = 0.0
      DYP(3,1) = 0.0
      DYP(3,2) = Y(3)*Y(1)
      DYP(3,3) = 0.0
      DYP(3,4) = 0.0
      DYP(3,5) = Y(3)*X(2)
      DYP(3,6) = 0.0
      DYP(3,7) = 1.0 + X(2)*Y(1)
      RETURN

C
C   DUMMY
C
  600 CONTINUE
      RETURN

C

```

```

C   GRADIENTS OF FITTING CRITERIA W.R.T. X AND Y
C
700 CONTINUE
    DFIT(1,1) = 0.0
    DFIT(1,2) = 0.0
    DFIT(1,3) = 0.0
    DFIT(1,4) = 1.0
    DFIT(1,5) = 0.0
    DFIT(1,6) = -1.0
    DFIT(1,7) = 0.0
    RETURN
C
C   GRADIENTS OF CONSTRAINTS
C
800 CONTINUE
    RETURN
C
C   GRADIENTS OF RIGHT-HAND SIDE OF EQUATIONS W.R.T. Y
C
900 CONTINUE
    DYP(1,1) = 1.0 + X(1)*Y(2) + X(2)*Y(3)
    DYP(1,2) = Y(1)*X(1)
    DYP(1,3) = Y(1)*X(2)
    DYP(2,1) = Y(2)*X(1)
    DYP(2,2) = 1.0 + X(1)*Y(1)
    DYP(2,3) = 0.0
    DYP(3,1) = Y(3)*X(2)
    DYP(3,2) = 0.0
    DYP(3,3) = 1.0 + X(2)*Y(1)
    RETURN
C
    END

```

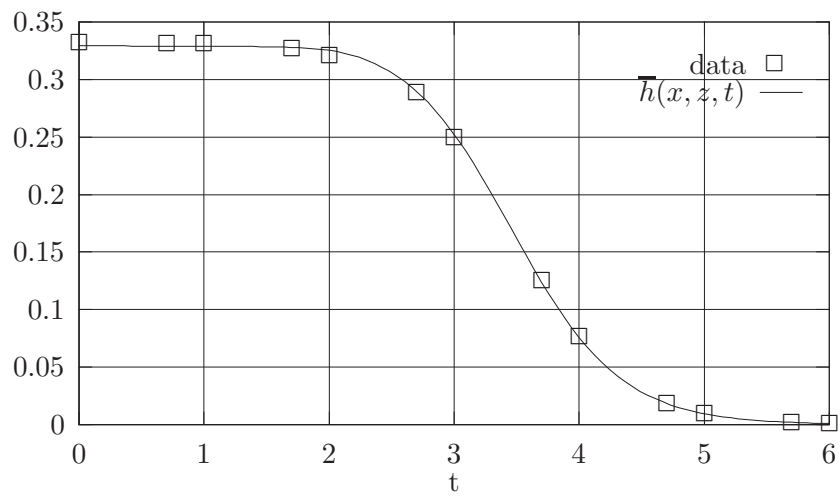


Figure 9.4: Data and Function Plot for Model DYN-EQ

9.2 PDEFIT: Parameter Estimation in Partial Differential Equations

Basically, PDEFIT is a double precision Fortran subroutine and fully documented by initial comments. Since most applications will probably execute the corresponding main program, we describe only the format of the input data and the usage of the subroutine required for model evaluation. More technical information can be retrieved from the initial comments of the main program, for example link files, parameters, common blocks etc., and from Dobmann and Schittkowski[117].

Among the data generated by PDEFIT, are result, report and plot files, that can be used for external programs evaluating these data. The format is described in detail among the initial comments of the Fortran code of PDEFIT. Numerical results are stored on a file with extension .RES, and are read by **EASY-FIT**^{ModelDesign} as soon as numerical execution is terminated. In case of a simulation run with a positive significance level, statistical data are written to a file with extension .STA in abbreviated form.

The code distributed together with **EASY-FIT**^{ModelDesign}, allows the input of model functions on a user provided file with name <MODEL>.FUN in a directory specified in the first line of PDEFIT.DAT. In this case, the input format is the PCOMP language must be used as outlined in the previous chapters. A specific advantage is the automatic evaluation of derivatives. There is no further compilation or link necessary and PDEFIT can be started immediately, as soon as both input files are created.

An input file named PDEFIT.DAT must contain the parameter estimation data, some problem information and optimization data in formatted form. The first 6 columns may contain an arbitrary string to identify the corresponding input row, if allowed by the format.

Line	Format	Name	Description
1	a80	FILE	Name of output files generated by PDEFIT. The string may begin with a path name, but must not contain an extension. Suitable extensions are selected by PDEFIT.
2	a6,4x,i5	MODEL	Problem name passed to subroutine SYSFUN for identifying data fitting models. The subsequent integer defines the general model structure:

Line	Format	Name	Description
			1 - system of partial differential equations 2 - system of partial differential algebraic equations
3	a70	INFO	Long information string for plot output.
4	a20	PROJECT	Plot output (first line of information block, e.g., project number).
5	a20	TEST	Plot output (second line of information block, e.g., measurement characterization).
6	a20	DATE	Plot output (third line of information block, e.g., date).
7	a10	Z-AXIS	Name for z -axis (spatial variable).
8	a10	Y-AXIS	Name for y -axis (value).
9	a10	T-AXIS	Name for t -axis (time).
10	a6,4x,2i5	NPAR NBPV	Number of parameters to be optimized, must be at least one. Number of variable break points, i.e., the last NBPV variables are treated as break points where integration is restarted.
11	a6,4x,i5	NPDE	Number of PDE's, must be at least one.
12	a6,4x,i5	NPAE	Number of algebraic equations.
13	a6,4x,i5	NCPLO,NCPLA	Numbers of coupled ordinary differential and algebraic equations.
14	a6,4x,i5	ICPLO	Formatted input of NCPLO lines where each line contains the line number, i.e., the discretization point, where the ODE is coupled to the PDE. The line with number 1 denotes the left boundary of the integration area.
15	a6,4x,i5	ICPLA	Formatted input of NCPLA lines where each line contains the line number, i.e., the discretization point, where the algebraic equation is coupled to the PDE. The line with number 1 denotes the left boundary of the integration area.

Line	Format	Name	Description
16	a6,4x,i5	NCPB	Number of area boundaries, must be an even number (since every area has a left and a right boundary).
17	a6,4x,i5	NRES	Number of constraints without bounds.
18	a6,4x,i5	NEQU	Number of equality constraints.
19	a6,4x, 2g20.4, i5	RT,RX,IX	Formatted input of NRES rows each containing two real numbers identifying the experimental time and spatial parameter values for which a constraint is to be supplied, and the corresponding line number. The order is arbitrary, but first the equality and subsequently the inequality constraints are to be defined. The data are rounded to the nearest actual time value.
20	a6,4x,2i5	NTIME, MPLOT	Number of time points, must be greater than 0. Logarithmic scaling of x-axis (MPLOT=1) or not (MPLOT=0).
21	a6,4x,i5	NFIT	Number of fitting criteria.
22	a6,4x,i5	IFIT	Formatted input of NFIT lines where each line contains the line number, i.e., the discretization point, where a fit criterion is defined. The line with number 1 denotes the left boundary of the integration area.
23	a6,4x,i5	NPLOT	Number of plot points to be computed by additional model function evaluations. Plots are generated by interpolation (linear or polynomial depending on graphics system). NPLOT may be zero if no plot data are required.
24	a6,4x,i5	NOUT	Output flag for PDEFIT. NOUT=0 - no generation of output files NOUT=1 - generation of output files
25	a6,4x,i5	DQUPOI	Number of points used to approximate the spatial derivatives for method of lines by polynomials of degree DQUPOI-1. DQUPOI must be odd and at least three for polynomial interpolation.

Line	Format	Name	Description
26	a6,4x,i5	APRMET	<p>Determines how to approximate spatial derivatives:</p> <p>APRMET=0: Polynomial approximation for u_x, u_{xx} in case of FLFLAG=1</p> <p>For FLFLAG > 1:</p> <p>APRMET=1: central differences for u_x and recursive application for u_{xx} (DQUPOI>2)</p> <p>APRMET=2: 5-point differences for u_x and recursive application for u_{xx} (DQUPOI>4)</p> <p>APRMET=3: 5-point differences for u_x and separately for u_{xx}</p> <p>APRMET=4: Forward differences for u_x</p> <p>APRMET=5: Backward differences for u_x</p> <p>APRMET=6: Individual selection of discretization formula for u_x, can be different for each state variable</p>
27	a6,4x,i5	FLFLAG	<p>Indicates existence and type of flux function:</p> <p>FLFLAG=0: No flux function defined</p> <p>FLFLAG=1: Flux function exists and is differentiated by chain-rule</p> <p>FLFLAG=2: Flux function exists and is discretized by the method defined below</p>
28	a6,4x,i5	APFLUX	<p>Parameter for choosing high resolution scheme. If $-1 \leq APFLUX \leq 1$, the one-parameter family of TVD schemes of Chakravarthy and Osher is used:</p>

Line	Format	Name	Description
			<p>APFLUX=-1.0 : upwind scheme</p> <p>APFLUX=-1/3 : no name</p> <p>APFLUX= 0.0 : Fromm scheme</p> <p>APFLUX= 1/3 : third-order upwind-biased scheme</p> <p>APFLUX= 1/2 : second-order scheme</p> <p>APFLUX= 1.0 : central difference scheme</p> <p>Alternatively, the user can choose between the following two high-resolution schemes:</p> <p>APFLUX= 2.0: first-order upwind scheme</p> <p>APFLUX= 3.0: second-order central differences</p> <p>If $11.0 < 99.1$, the system of PDE's is supposed to consist of advection equations of the form $\partial u / \partial t = \partial f(p, u) / \partial x$ plus inhomogeneous part to be discretized by the ENO method with APFLUX= IJ:</p> <p>I : Approximating polynomial order for state variable u at cell wall by Marquina's rule</p> <p>J : Approximating polynomial order for numerical flux by ENO-Roe rule</p>
29	a6,4x,3i5	METHOD, NORM, NUMGRA	<p>Choice of analysis or optimization algorithm:</p> <p>METHOD=0 - Simulation</p> <p>METHOD=1 - Call of DFNLP (Schittkowski [429])</p> <p>METHOD=2 - dummy</p> <p>NORM determines the data fitting norm.</p> <p>NORM=1 = L_1 norm, sum of absolute residuals</p> <p>NORM=2 = L_2 norm, sum of squared residuals</p>

Line	Format	Name	Description
			<p>NORM=3 = L_∞ norm, maximum of absolute residuals</p> <p>L1- and maximum-norm are applicable only for a simulation run or a data fitting run with DFNLP.</p> <p>NUMGRA must be set for gradient evaluation.</p> <p>NUMGRA=-1 - analytical derivatives available</p> <p>NUMGRA=0/1 - forward differences</p> <p>NUMGRA=2 - two-sided differences</p> <p>NUMGRA=3 - 5-point difference formula</p>
30	a6,4x,i5	OTPT1	Maximum number of iterations for the chosen optimization algorithm. In case of a simulation, run, the desired significance level 1, 5 or 10 is to be inserted.
31	a6,4x,i5	OTPT2	<p>Additional parameter for chosen optimization algorithm:</p> <p>METHOD=1 - maximum number of line search steps</p>
32	a6,4x,i5	OTPT3	Output level of optimization algorithm chosen. The output is directed to a file with extension HIS. Only the residuals are displayed on screen.
33	a6,4x,g10.4	OPTE1	<p>Tolerance for chosen optimization algorithm:</p> <p>METHOD=0 - tolerance for rank determination</p> <p>METHOD=1 - final termination tolerance</p>
34	a6,4x,g10.4	OPTE2	<p>Tolerance for chosen optimization algorithm:</p> <p>METHOD=1 - expected size of residual</p>
35	a6,4x,g10.4	OPTE3	<p>Tolerance for chosen optimization algorithm:</p> <p>dummy</p>
36	a6,4x,i5	ODEP1	Parameter for selection of differential equation solver:

Line	Format	Name	Description
			ODEP1=1 - dummy ODEP1=2 - dummy ODEP1=3 - dummy ODEP1=4 - RADAU5 (implicit Runge-Kutta, order 5) ODEP1=5 - dummy ODEP1=6 - dummy ODEP1=7 - TVDRK (explicit Runge-Kutta, order 5, for systems of advection equations without inhomogeneous term)
37	a6,4x,i5	ODEP2	Order of used method (only for implicit methods): ODEP2=0 - no derivatives ODEP2=1 - derivatives of right hand side supplied
38	a6,4x,i5	ODEP3	Approximate number of correct digits when gradients are evaluated numerically by forward differences. If set to 0, the tolerance is internally computed.
39	a6,4x,5i5	ODEP4	Bandwidth of Jacobian of right-hand side (only for implicit solver), must be smaller than number of ODE's of discretized system. In case of ODEP4=0, usage of full matrix is assumed.
40	a6,4x,g10.4	ODEE1	Final termination accuracy for ODE-solver with respect to the relative global error.
41	a6,4x,g10.4	ODEE2	Final termination accuracy for ODE-solver with respect to the absolute global error. In case of TVDRK, the parameter is used to pass a factor for reducing the stepsize if the CFL condition is not satisfied (>1 in this case!).
42	a6,4x,g10.4	ODEE3	Tolerance for solving differential equation: initial step-size.

Line	Format	Name	Description
43	a6,4x,g10.4	XSTART	Value of the spatial component at the leftmost boundary
44	a6,4x,g10.4, 5i5 10x,3i5		<p>Formatted input of $NCPB/2 \dots (NPDE + 1)$ lines.</p> <p>Each headline contains</p> <ul style="list-style-type: none"> - the name of the area, - the spatial size of the area, - the number of discretization points in the area, <p>whereas the following NPDE lines contain for each PDE</p> <ul style="list-style-type: none"> - status of left boundary condition, - status of right boundary condition, - spatial derivative approximation (APRMET=6). <p>The boundary status is:</p> <p>0 - no boundary condition</p> <p>1 - Dirichlet boundary condition</p> <p>2 - Neumann boundary condition</p> <p>Possible spatial derivative approximations are</p> <p>0 - order taken from APRMET</p> <p>1 - central differences for u_x and recursive application for u_{xx} (DQUPOI>2)</p> <p>2 - 5-point differences for u_x and recursive application for u_{xx} (DQUPOI>4)</p> <p>4 - Forward differences for u_x</p>

Line	Format	Name	Description
			5 - Backward differences for u_x
45	a6,4x,3g20.8		<p>Formatted input of NPAR lines each containing three real numbers for</p> <ul style="list-style-type: none"> - lower bound for estimated parameter - starting value for estimated parameter - upper bound for estimated parameter <p>If the file '*.RES' contains the results of a previous run, then the corresponding parameter values are read in and replace the given ones, i.e., the PAR-values.</p>
46	a6,4x,i5	SCALE	<p>Scale for weight factors:</p> <ul style="list-style-type: none"> 0 - no additional scaling of weights 1 - divide each weight factor by square root of sum of squared measurement values -1 - divide each weight factor by absolute measurement value -2 - divide each weight factor by squared measurement value.
47	*		<p>Unformatted input of NTIME lines for $i = 1 \dots NTIME$ with the following data:</p> <p>t_i - i-th measurement time, not smaller than zero</p> <p>y_i^k, w_i^k - measured data, i.e., experimental output, and individual weight factor for measurement number k, $k = 1, \dots, NMEAS$.</p>
48	a6,4x,i5	INTEG	Flag for evaluation of integrals with respect to spatial variable X (=1) or not (=0).

Line	Format	Name	Description
49	a6,4x,i5	ORDER	Order of partial differential equation variable X. ORDER = 1: Only first derivatives (hyperbolic) ORDER = 2: First and second derivatives (parabolic)
50	a6,4x,i5	NLPIP	Print flag for inner nonlinear equation solver NLPQLP executed to get consistent initial values in case of additional algebraic equations: NLPIP = 0 : no output NLPIP = 1 : only final output NLPIP = 2 : one line per iteration NLPIP = 3 : extended output every iteration
51	a6,4x,i5	NLPMI	Maximum number of iterations for NLPQLP (e.g. 50)
52	a6,4x,g10.4	NLPAC	Convergence tolerance for NLPQLP (e.g. 1.d-12)
53	a6,4x,i5	NBPC	Number of constant break points where integration is restarted with initial tolerances. Constant break points are permitted only if NBPV=0.
54	*		Unformatted input of NBPC rows each containing one time value in increasing order that represents a break point of the right-hand side of a PDE.

A user has the alternative option to implement all model functions in form of Fortran code, and to link his own module to the object file of PDEFIT. Information about the remaining files to be linked in this case, is found among the initial comments of the file PDEFIT.FOR that contains the main program. The model data, i.e., fitting criterion, system equations, bounds and initial values, must be provided by a user-defined subroutine called SYSFUN:

```

SUBROUTINE SYSFUN(NPAR,MAXPAR,NPDE,MAXPDE,NCPL,MAXCPL,NMEA,
/
MAXMEA,NRES,MAXRES,PAR,U,UO,UX,UXX,UP,V,VO,
/
VP,C,CX,FIT,G,DG,X,T,IAREA,LEFT,RIGHT,IFLAG,
/
FLUX,FLUXX,FLUXU,FLUXUX,FLUXPX)

```

The meaning of the arguments is as follows:

NPAR	Number of parameters in array PAR.
MAXPAR	Dimensioning parameter, must be greater or equal to NPAR.
NPDE	Number of functions on right-hand side of partial differential equations.
MAXPDE	Dimensioning parameter, must be greater or equal to NPDE.
NCPL	Number of coupled differential algebraic equations.
MAXCPL	Dimensioning parameter, must be greater or equal to NCPL.
NMEA	Number of measurement sets.
MAXMEA	Dimensioning parameter, must be greater or equal to NMEA.
NRES	Number of constraint functions in the parameter estimation problem.
MAXRES	Dimensioning parameter, must be greater or equal to NRES.
PAR(MAXPAR)	When calling 'SYSFUN', PAR contains the NPAR coefficients of the actual variables to be estimated. PAR is not allowed to be altered within the subroutine.
U(MAXPDE)	When calling 'SYSFUN', U contains the coefficients of the partial differential equations on the right-hand side for the spatial discretization point X, as described subsequently. U is not allowed to be altered within the subroutine.
U0(MAXPDE)	Function values to be evaluated in the case of 'IFLAG=3' for initial values of PDE's.
UX(MAXPDE)	When calling 'SYSFUN', UX contains the coefficients of the first derivatives of the solution of the partial differential equations for the spatial discretization point X and the transition conditions. UX is not allowed to be altered within the subroutine.
UXX(MAXPDE)	When calling 'SYSFUN', UXX contains the coefficients of the second derivatives of the solution of the partial differential equations for the spatial discretization point X and the transition conditions. UXX is not allowed to be altered within the subroutine.
UP(MAXPDE)	Function values to be evaluated in the case of 'IFLAG=2' for right-hand side of PDE's.
V(MAXCPL)	When calling 'SYSFUN', V contains the coefficients of the coupled differential algebraic equations on the right-hand side. V is not allowed to be altered within the subroutine.
V0(MAXCPL)	Function values to be evaluated in the case of 'IFLAG=5' for initial values of the coupled differential algebraic equations.
VP(MAXCPL)	Function values to be evaluated in the case of 'IFLAG=4' for right-hand side of the coupled ODE'S.

C(MAXPDE)	Array with boundary or transition values in case 'IFLAG=6' or 'IFLAG=7'. The vector has to contain the computed values of the partial differential equations at an internal transition point X or at an external boundary point, when leaving the subroutine.
CX(MAXPDE)	Array with boundary or transition values in case of 'IFLAG=7'. The vector must contain the computed values of the derivatives of the differential equations for the spatial variable X at an internal transition point X or at an external boundary point, when leaving the subroutine.
FIT(MAXMEA)	Function values to be evaluated in the case of 'IFLAG=8' for the NMEA fitting conditions of the parameter estimation problem.
G(MAXRES)	Function values to be evaluated in case of 'IFLAG=9' for constraints in the parameter estimation problem.
DG(MAXRES, MAXPAR)	Gradients of constraints with respect to parameters to be estimated, dummy parameter.
X	Value of the spatial component X at an actual discretization point.
T	Time variable T.
IAREA	Integer variable to inform the user about the actual area.
LEFT	Logical to inform the user about the boundary location. The variable is TRUE if and only if the actual X-value defines the left boundary.
RIGHT	Logical to inform the user about the boundary location. The variable TRUE if and only if the actual X-value defines the right boundary.
IFLAG	Flag defining the desired type of calculation. 0 - Execution of SYSFUN before requiring function or gradient values, e.g., for preparing common's. 1 - Evaluate flux functions of PDE's and store results in FLUX. 2 - Evaluate right-hand side of PDE's and store results in UP. 3 - Evaluate initial values of PDE's and store results in U0. 4 - Evaluate right-hand side of coupled ODE's followed by coupled algebraic equations and store results in VP. 5 - Evaluate initial values of coupled ODE's followed by initial values for algebraic equations and store results in V0. 6 - Evaluate transition functions for PDE's. 7 - Evaluate transition derivatives for PDE's.

	8 - Evaluate fitting criteria and store results in FIT.
	9 - Evaluate constraints and store values in G.
	11 - Evaluate the partial derivatives of flux functions subject to U, UX and X and store results in FLUXU, FLUXUX, and FLUXPX.
FLUX(MAXPDE)	When calling SYSFUN, FLUX contains the coefficients of the flux functions of the partial differential equations.
FLUXX(MAXPDE)	When calling SYSFUN, FLUXX contains the coefficients of the first derivatives of the flux functions.
FLUXU(MAXPDE, MAXPDE)	Has to get the derivatives of the flux functions for U when calling the subroutine with 'IFLAG=11'.
FLUXUX(MAXPDE, MAXPDE)	Has to get the derivatives of the flux functions for UX when calling the subroutine with 'IFLAG=11'.
FLUXP(MAXPDE)	Has to get the derivatives of the flux functions for PAR when calling the subroutine with 'IFLAG=11'.

If INTEG is set to 1, PDEFIT computes the integral

$$\int_{x_{j-1}^a}^{x_j^a} u^i(p, x, t) dx$$

where $j = 1, \dots, n_t$ and $i = 1, \dots, n_p$. The integral is evaluated by Simpson's rule and denoted by

SIMPSN(I, J)

in the PCOMP language. In case of a Fortran implementation of the model functions, the same value can be retrieved from a public common block kept in the include file PDEFIT.INC, under the name AINTEG(I, J). Note that also access to the complete solution array $u^i(p, x_k, t)$ at the discretization points x_k is also possible. The corresponding array is denoted by USOL(I, K). The time value is either a measurement value or an intermediate value needed for generating plot data.

Example 9.5 (HEAT) *We consider now Example 6.6 again, a simple heat conduction model found in Schiesser [420], where Fourier's first law for heat conduction leads to the equation*

$$u_t = u_{xx} \tag{9.5}$$

defined for $0 < t \leq 0.5$ and $0 < x < 1$. Boundary conditions are

$$u(0, t) = u(1, t) = 0 \tag{9.6}$$

for $0 \leq t \leq 0.5$ and the initial values are

$$u(x, 0) = \sin\left(\frac{\pi x}{L}\right) \tag{9.7}$$

for $0 < x < 1$ and $0 < L \leq 1$. In addition, we are interested in the total amount of heat at the surface $x = 0$

$$\dot{v} = -K \cdot \frac{\pi}{L} \cdot e^{-\frac{\pi^2}{L^2} \cdot t} \quad (9.8)$$

with initial heat

$$v_0 = \frac{K \cdot L}{\pi}$$

Function v serves also as our fitting criterion. Then the corresponding Fortran code is to be implemented as follows:

```

      SUBROUTINE SYSFUN(NPAR,MAXPAR,NPDE,MAXPDE,NCPL,MAXCPL,
/
/          NMEA,MAXMEA,NRES,MAXRES,PAR,U,UO,UX,
/          UXX,UP,V,VO,VP,C,CX,FIT,G,DG,X,T,IAREA,
/          LEFT,RIGHT,IFLAG,FLUX,FLUXX,FLUXU,
/          FLUXUX,FLUXPX)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION PAR(MAXPAR),U(MAXPDE),UO(MAXPDE),UX(MAXPDE),
/          UXX(MAXPDE),UP(MAXPDE),V(MAXCPL),VO(MAXCPL),
/          VP(MAXCPL),C(MAXPDE),CX(MAXPDE),FIT(MAXMEA),
/          G(MAXRES),DG(MAXRES,MAXPAR),
/          FLUX(MAXPDE),FLUXX(MAXPDE),
/          FLUXU(MAXPDE,MAXPDE),FLUXUX(MAXPDE,MAXPDE),
/          FLUXPX(MAXPDE)
      LOGICAL LEFT,RIGHT
      DOUBLE PRECISION K,L
C
      IF (IFLAG.EQ.0) RETURN
C
C   SET PARAMETERS
C
      L  = PAR(1)
      K  = PAR(2)
      PI = 3.1415926535
C
C   BRANCH W.R.T. IFLAG
C
      GOTO (100,200,300,400,500,600,700,800,900,1000,1100) IFLAG
      RETURN
C
C   EVALUATION OF FLUX FUNCTION
C
100 CONTINUE
      RETURN
C
C   RIGHT-HAND SIDE OF PDE'S
C
200 CONTINUE

```

```

        UP(1) = UXX(1)
        RETURN
C
C   INITIAL VALUES OF PDE'S
C
      300 CONTINUE
        UO(1) = DSIN(PI*X/L)
C
C   RIGHT-HAND SIDE OF ODE'S
C
      400 CONTINUE
        VP(1) = -K*PI/L*DEXP(-(PI/L)**2*T)
        RETURN
C
C   INITIAL VALUES OF ODE'S
C
      500 CONTINUE
        VO(1) = K*L/PI
        RETURN
C
C   BOUNDARY FUNCTIONS
C
      600 CONTINUE
        IF (LEFT) C(1)=0.0
        IF (RIGHT) C(1)=0.0
        RETURN
C
C   BOUNDARY GRADIENTS
C
      700 CONTINUE
        RETURN
C
C   FITTING CRITERIA
C
      800 CONTINUE
        FIT(1) = V(1)
        RETURN
C
C   CONSTRAINTS
C
      900 CONTINUE
        RETURN
C
C   GRADIENTS OF CONSTRAINTS
C
     1000 CONTINUE
        RETURN
C
C   GRADIENTS OF FLUX FUNCTIONS

```

```

C
  1100 CONTINUE
      RETURN
C
      END

```

Parameters to be estimated, are L and K . Measurements are simulated subject to $L = 1$, $K = 2$ at the spatial coordinates 0, 0.1, 0.2, 0.3, 0.4, and 0.5. The corresponding input file is the following one, where DFNLP is called for parameter estimation and RADAU5 for solving the discretized system of ordinary equations. Starting values for DFNLP are $L = 2$ and $K = 3$.

```

C:\EASYFIT\PROBLEMS\HEAT
HEAT      6
Heat conduction
One compartment
Test
18.01.1995
deg
mm
min
NPAR  =      2      0
NPDE  =      1
NPAE  =      0
NCPL  =      1      0
ICPL  =      1
NCPB  =      2
NRES  =      0
NEQU  =      0
NTIME =      6      0
NFIT  =      1
IFIT  =      1
NPLOT =     20
NOUT  =      1
DQUPOI=      3
APRMET=      0
FLUX  =      0
APRFLX=      0.0
OPTMET=      1
OPTP1 =     100
OPTP2 =      8
OPTP3 =      2
OPTE1 =     1.0E-7
OPTE2 =     1.0E-1
OPTE3 =     1.0E+2
ODEP1 =      4
ODEP2 =      0
ODEP3 =      6

```

```

ODEP4 =      0
ODEE1 =     1.0E-5
ODEE2 =     1.0E-5
ODEE3 =     1.0E-5
XSTART=      0.0
COMP1      1.0  11
           1    1    0
L           0.0           2.0          10.0
K           0.0           3.0          10.0
SCALE =      0
0.0  0.636619      100.0
0.1  0.237273      100.0
0.2  8.84335E-2    100.0
0.3  3.29598E-2    100.0
0.4  1.22844E-2    100.0
0.5  4.57849E-3    100.0
INTEG =      0
ORDER =      2
NLPIP =      0
NLPMI =      0
NLPAC =      0.0
NBPC  =      0

```

DFNLP terminates after 22 steps with a residual of $0.22 \cdot 10^{-9}$, see Figure 9.6, which was evaluated for a finer grid with 25 discretization points.

More examples how to formulate model functions and boundary conditions are found in Dobmann and Schittkowski [117].

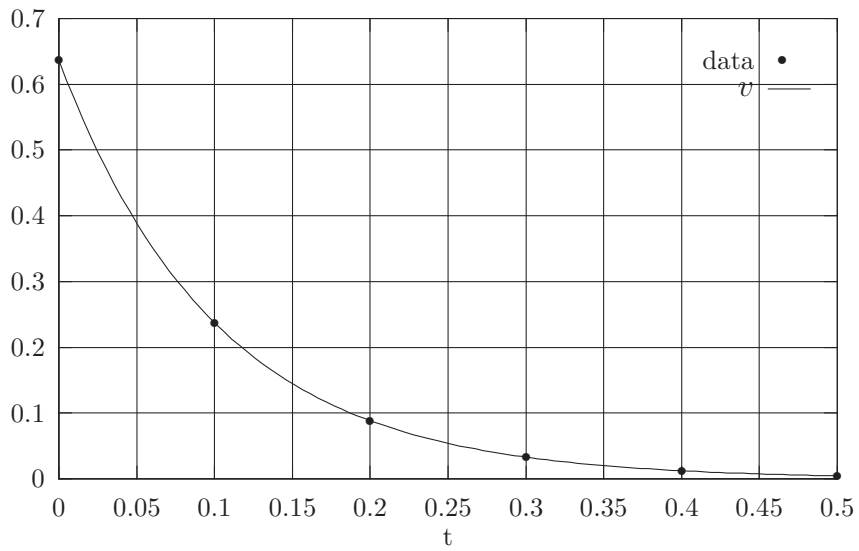


Figure 9.5: Final Trajectory for Heat Conduction Problem

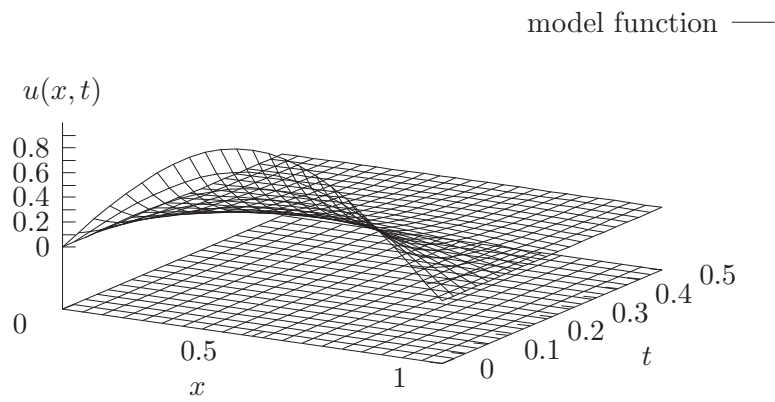


Figure 9.6: Surface Plot for Heat Conduction Problem

Chapter 10

Test Examples

The reason for attaching a comprehensive collection of test problems is to offer the possibility to try out different discretization procedures, differential equation solvers, and data fitting algorithms. The problems can be used for selecting a reference problem when trying to implement own dynamical models, or to test the accuracy and efficiency of numerical algorithms, for example for comparisons with other methods.

In many cases, parameter estimation problems are found in the literature or are based on cooperation with people from other academic or industrial institutions. In many other cases, however, differential equations are taken from research articles about numerical simulation algorithms, and are adapted to construct a suitable data fitting test problem. Thus, some model equations do not coincide exactly with those given in the corresponding references and the numerical solution is sometimes different from the one found in the reference.

We summarize a few characteristic data and the application background of the test problems that are available on the CD-ROM, from where further details can be retrieved. Besides of problem name and some figures characterizing problem size, we present also information how measurement data are obtained,

- E - Experimental data from literature or private communication,
- S0 - simulation without error,
- S05 - simulation with uniformly distributed error of 0.5 %,
- S1 - simulation with uniformly distributed error of 1 %,
- S5 - simulation with uniformly distributed error of 5 %,
- S10 - simulation with uniformly distributed error of 10 %,
- S50 - simulation with uniformly distributed error of 50 %,
- N1 - simulation with normally distributed error, $\sigma = 1$,
- N5 - simulation with normally distributed error, $\sigma = 5$,
- X - comparison with exact solution,
- ED - experimental design,
- none - no experimental data set, for example least squares test problem.

The difference between *simulated* and *experimental* data is that exact parameter values are

known in the first case. Besides of a large collection of problems with practical experimental data, there are also a few others where the data are constructed, for instance in case of some optimal control problems, or are determined more or less *by hand*. In many other situations, the exact solution of the differential equation is known and used to simulate experimental data. These test examples can be used to check the accuracy of discretization formulae or ODE solvers.

Moreover, we show some references in the column headed by *ref*, from where further details can be retrieved. Either the data fitting problem is described in detail, or at least the mathematical background of the model is outlined. In case of an empty entry, the model is provided by private communication and not published somewhere else, or a related reference is unknown to the author.

To summarize, we offer test problems for the following model classes:

explicit model functions	:	243
Laplace transforms	:	10
steady state equations	:	41
ordinary differential equations	:	575
differential algebraic equations	:	62
partial differential equations	:	325
partial differential algebraic equations	:	44
sum	:	1,300

10.1 Explicit Model Functions

We proceed from r measurement sets of the form

$$(t_i, c_j, y_{ij}^k) , \ i = 1, \dots, l_t, \ j = 1, \dots, l_c, \ k = 1, \dots, r$$

with l_t time values, l_c concentration values, and $l = l_t l_c r$ corresponding measured experimental data. Moreover, we assume that l weights w_{ij}^k are given. However, weights can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for functions for which experimental data do not exist. Thus, the subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation.

Usually, we proceed from the L_2 - or Euclidean norm to formulate a parameter estimation problem of the form (2.16),

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, t_i, c_j) - y_{ij}^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0 , \ j = 1, \dots, m_e , \\ & g_j(p) \geq 0 , \ j = m_e + 1, \dots, m_r , \\ & p_l \leq p \leq p_u , \end{aligned}$$

where we assume that fitting criteria $h_k(p, t, c)$, $k = 1, \dots, r$, and constraints $g_j(p)$, $j = 1, \dots, m_r$, are continuously differentiable functions subject to p . The model function $h(p, t, c) = (h_1(p, t, c), \dots, h_r(p, t, c))^T$ does not depend on the solution of an additional dynamical system and can be evaluated directly from a given parameter vector p that is to be estimated at given time and concentration values t and c . All explicit test problems are listed in Table B.1.

Table B.1. Explicit Model Function

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
2INDVARS	4	10	0	0	Problem with 2 independent model parameters		S5
2VALLEYS	2	4	0	0	Academic test problem with two local minima	[458]	E
3MODVARS	9	602	0	0	A demo problem with three independent model variables		E
4BAR_LNK	4	24	2	0	Design of a four bar linkage		X
ADL_CSTR	52	50	30	30	Steady-state adiabatic CSTR with irreversible first order reaction and errors in variables	[251]	E
APPRX3	6	10	0	0	Rational approximation of data	[28]	S5
ASPHER_1	6	53	2	0	Aspheric lens shape, lenticular measurements (1st data set)		E
ASPHER_2	6	27	2	0	Aspheric lens shape, lenticular measurements (2nd data set)		E
ASPHER_3	6	53	2	0	Aspheric lens shape, lenticular measurements (3rd data set)		E
ASPHER_4	6	53	2	0	Aspheric lens shape, lenticular measurements (4th data set)		E
ATROP_EX	4	24	0	0	Atropin-chase binding, linear model		E
BENNETT5	3	154	0	0	Superconductivity magnetization modeling (NIST study)		E
BIRDMILL	3	14	0	0	Non-identifiability	[458], [42]	S5
BOGGS2	3	3	1	1	Test problem of Boggs with rank deficient Jacobian at start and equality constraints	[428]	none
BOGGS8	5	3	2	2	Test problem of Boggs with rank deficient Jacobian at solution and equality constraints	[428]	none
BOXBOD	2	6	0	0	Biochemical oxygen demand (NIST study)	[59]	E
BURGER_W	3	50	0	0	Explicit solution of Burger's equation with eps=0.0005		N1
CAT_SEP	5	5	0	0	Catalysator separation problem		E
CEMENT	28	63	0	0	Hardening of cement	[101]	E
CENSUS	2	13	0	0	US census over years 1790 to 1900	[182]	E
CHEMOSTA	3	69	0	0	Steady-state chemostat	[126]	S5
CHWIRUT1	3	214	0	0	Ultrasonic reference block (NIST study)		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
CHWIRUT2	3	54	0	0	Ultrasonic reference block (NIST study)		E
COMBPROP	13	13	0	0	Combustion of propane	[222]	none
CYC_COM1	2	10	0	0	Stability of cyclodextrin complexes (MeBCd)		E
CYC_COM2	2	10	0	0	Stability of cyclodextrin complexes (HEtBCd)		E
DA_X	4	72	0	0	MDI simulation		E
DANWOOD	2	6	0	0	Energy radiated from a carbon filament lamp (NIST study)	[101]	E
DCA_CON	2	18	0	0	Decline curve analysis of reservoirs, constant percentage decline	[481]	E
DCA_HAR	2	18	0	0	Decline curve analysis of reservoirs, harmonic decline	[481]	E
DCA_HYP	2	18	0	0	Decline curve analysis of reservoirs, hyperbolic decline	[481]	E
DEGRAD3	3	20	0	0	Microbial degradation with hydrolysis and photolysis, explicit model		E
DEGRAD4	3	20	0	0	Microbial degradation with hydrolysis and photolysis, parallel version, explicit model		E
DENSITY	6	37	0	0	Density data of a liquid crystal		E
DFBLASER	4	4	0	0	Dynamic characteristics of a semi-conductor DFB laser		none
DFE1	8	9	0	0	Explicit test function with local solutions, cycling model function etc.		S0
DFE2	7	15	1	1	Explicit test function with time-dependent model change		S5
DIFFREAX	6	292	0	0	Diffusion and reaction in solid phase, explicit formulation		E
DISS_TAB	2	4	0	0	Dissolution of tablets		E
DNS	3	30	1	0	Feulgen-hydrolysis of DNS, biochemical reaction	[384]	E
DOAS	21	100	0	0	Differential optical spectral absorption		S0
DRUG	2	14	1	0	Bolus intravenous injection of a drug	[182]	E
E_FIT	3	12	0	0	Rational-exponential data fitting		E
ECKERLE4	3	35	0	0	Circular interference transmittance (NIST study)		E
ELA_TUBX	3	40	0	0	Waves propagating in a liquid-filled elastic tube (KDVb equation)	[241], [521]	S5
ENSO	9	168	0	0	Atmospheric pressure differences (NIST study)	[244]	E
ENZREAC	4	13	0	0	Enzyme reaction, rational approximation		E
EW_WAVEX	2	48	0	0	Wave propagation in media with nonlinear steepening and dispersion	[201]	S5
EXP_FIT1	2	28	0	0	Exponential data fitting		S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
EXP_FIT2	7	33	0	0	Exponential data fitting, explicit solution of linear ODE		S0
EXP_FIT3	2	27	0	0	Exponential data fitting		E
EXP_FIT4	5	19	0	0	Exponential data fitting		E
EXP_FIT5	10	20	0	0	Exponential data fitting		E
EXP_FIT6	2	4	0	0	Exponential data fitting		E
EXP_FIT7	2	11	0	0	Explicit test function		S0
EXP_P1	2	3	0	0	Test example: trigonometric functions, overdetermined	[466]	none
EXP_P2	2	3	0	0	Test example: rational functions, overdetermined	[466]	none
EXP_P4	20	20	0	0	Test example: linear functions	[466]	none
EXP_P5	2	2	0	0	Test example: polynomial functions	[466]	none
EXP_P6	2	2	0	0	Test example: polynomial functions	[466]	none
EXP_P7	2	1	0	0	Test example: polynomial functions, underdetermined	[466]	none
EXP_SMP_L	2	80	0	0	Single term exponential model, large errors in data		S50
EXP_TEST	4	200	0	0	Overlap of two exponential terms		S5
EXP_TST	4	20	0	0	Explicit test function, sum of two exponential terms		S0
EXP2TERM	5	20	0	0	Two-exponential model		E
F_DEHYDE	4	465	0	0	Colorimetric determination of formaldehyde	[367]	N5
GAMMAS	7	27	0	0	Analysis of a gamma spectrum		E
GAUSS	16	401	0	0	Distribution of points in Cartesian space fitted to linear combination of Gaussian functions		E
GAUSS_3D	6	29	0	0	Distribution of points in Cartesian space fitted to linear combination of Gaussian functions		E
GAUSS_AR	16	256	0	0	Fitting of data to eight gaussians		E
GAUSS1	8	250	0	0	Two well-separated Gaussians (NIST study)		E
GAUSS2	8	250	0	0	Two slightly-blended Gaussians (NIST study)		E
GAUSS3	8	250	0	0	Two strongly-blended Gaussians (NIST study)		E
GEAR	6	33	2	0	Gear with six parts		E
GEO_PROB	3	1	2	2	Maximum distance from origin to intersection of ellipsoid with hyperboloid	[304]	none

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
GLU_RATE	4	13	0	0	In-vivo glucose turnover rate		E
GO1	5	15	0	0	Test problem for global optimization	[382]	none
GO2	5	15	0	0	Test problem for global optimization	[382]	none
GO3	3	3	0	0	Test problem for global optimization (system of 3 equations)	[382]	none
GO4	3	1	3	3	Test problem for global optimization (system of 3 equations)	[382]	none
GO5	1	3	0	0	Test problem for global optimization (system of 3 equations)	[382]	none
GO6	1	1	1	1	Test problem for global optimization (system of 3 equations)	[382]	none
HAHN1	7	236	0	0	Thermal expansion of copper (NIST study)		E
HEAT_XX	2	99	0	0	Linear diffusion with constant parameters, exact solution		X
HEMISPH	3	100	0	0	Center of gravity of hemisphere		none
HEXA	5	149	0	0	Longitudinal sound velocity as function of propagation direction		E
HIMMELBD	2	2	0	0	System of two equation with local L2-solution		X
HYDENZYM	5	41	0	0	Hydrophobe enzymes and substrates		S0
HYP_ELA	8	231	17	0	Incompressible isotropic elastic strain-energy function		S1
ILLCOND	100	100	0	0	Ill-conditioned test function, many parameters		X
INFINITE	3	21	0	0	Infinitely many solutions		S0
INTEG_X	3	25	0	0	Population dynamics	[376]	S5
INTPOL	3	10	1	0	Interpolation routines, also non-continuous, non-smooth formulation		S0
IO3EXP	3	201	0	0	Input-output three-exponential model	[504]	N1
IRON_CC1	2	8	0	0	Iron carbochlorination		E
IRON_CCH	2	17	0	0	Iron carbochlorination		E
ISOMER_X	5	40	0	0	Thermal isomerization of alpha-pinene to dipentene	[399],[532] [58], [458]	E
KIRBY2	5	151	0	0	Scanning electron microscope (NIST study)		E
KS_FUNC	3	402	0	0	Kreiselmeier-Steinhauser function		X
LANCZOS1	6	24	0	0	Exponential nonlinear regression (NIST study)	[268]	E
LANCZOS2	6	24	0	0	Exponential nonlinear regression (NIST study)	[268]	E
LANCZOS3	6	24	0	0	Exponential nonlinear regression (NIST study)	[268]	E
LIFE	2	16	0	0	Nonlinear regression		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
LIN_CMP1	7	10	3	2	Linear compartments with bolus administration, single dose	[217]	S5
LIN_CMP2	9	54	3	3	Linear compartments with multidose administration (extravascular)	[217]	S5
LIN_CMP3	3	19	0	0	Multidose administration (extravascular)	[217]	E
LIN_HC_X	3	165	0	0	Linear heat conduction	[3]	S5
LIN_KIN	6	32	0	0	Linear pharmacokinetic model with 3-compartments and lag time		S5
LIN_MOD	12	30	0	0	Linear data fitting with errors in time values	[509]	E
LIN_VIS	22	84	9	0	Linear-viscoelastic material law in frequency domain		X
LIQUID	9	34	1	1	Solution in soil		E
LKIN_X	3	28	0	0	Linear compartment model with bolus application and single dose		E
LKIN_X3	2	78	0	0	Linear compartment model with bolus application and three doses		S5
MAC_ECO	6	186	0	0	Macroeconomic time series of currency notes in circulation	[497]	N1
MARKET	7	100	0	0	Dynamic economic market		E
MARKET_C	7	750	0	0	Dynamic economic market, 750 data		E
MGH09	4	11	0	0	Rational nonlinear regression (NIST study)	[341]	E
MGH10	3	16	0	0	Exponential nonlinear regression (NIST study)	[341]	E
MGH17	5	33	0	0	Exponential nonlinear regression (NIST study)	[341], [368]	E
MICHMENT	2	12	0	0	Michaelis-Menten kinetics	[458], [543]	E
MIME	24	172	0	0	Revolution lines		E
MISRA1A	2	14	0	0	Monomolecular adsorption (NIST study)		E
MISRA1B	2	14	0	0	Monomolecular adsorption (NIST study)		E
MISRA1C	2	14	0	0	Monomolecular adsorption (NIST study)		E
MISRA1D	2	14	0	0	Monomolecular adsorption (NIST study)		E
MIX_PAT1	2	38	1	0	Mixing pattern inside a polymerization reactor		E
MIX_PAT2	3	33	0	0	Mixing pattern inside a polymerization reactor		E
MIX_PAT3	1	27	0	0	Mixing pattern inside a polymerization reactor		E
MIX_PAT4	3	28	0	0	Mixing pattern inside a polymerization reactor		E
MONOD	4	10	0	0	Monod-Wymnan-Changueux kinetic equation	[458], [396]	S5
MORTALTY	2	9	0	0	Mortality rate by Gompertz function	[224]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
NELSON	3	128	0	0	Analysis of performance degradation data (NIST study)	[355]	E
NLP_E5	4	41	0	0	Testing nonlinear programs with very many constraints	[430],[200]	none
NLP_L5	7	1001	8	2	Testing nonlinear programs with very many constraints	[430],[303]	none
NLP_P1	3	3	50	0	Testing nonlinear programs with very many constraints	[430],[501]	none
NLP_P3	3	3	50	0	Testing nonlinear programs with very many constraints	[430],[501]	none
NLP_U3	5	251	0	0	Testing nonlinear programs with very many constraints	[430],[303]	X
OAT1	4	6	0	0	Bio-mass of oats	[403]	E
OAT2	3	6	0	0	Bio-mass of oats	[403]	E
OPT_KINX	6	60	2	0	Linear kinetics with variable switching times (optimal control problem)		none
OSCILL_S	16	50	0	0	Oscillating system with exact known solution	[562]	S0
OSCILL_X	16	50	0	0	Oscillating system	[562]	E
PARID120	3	121	0	0	Parameter identification model, 120 normally distributed experimental values		N1
PARID15	3	16	0	0	Parameter identification model, 15 normally distributed experimental values		N1
PARID30	3	31	0	0	Parameter identification model, 30 normally distributed experimental values		N1
PARID60	3	61	0	0	Parameter identification model, 60 normally distributed experimental values		N1
PERM1	4	4	0	0	Global optimization test problem (n=4, beta=50)		none
PERM2	4	4	0	0	Global optimization test problem (n=4, beta=0.5)		none
PERM3	10	10	0	0	Global optimization test problem (n=10, beta=1)		none
POL_APP	14	19	1	1	Polynomial approximation for computing axial forces		E
POL_MOD	14	30	0	0	Polynomial data fitting with errors in time values	[509]	E
POLARI	6	290	0	0	Fluorescence of polarization filter		E
POLYBU_X	5	68	0	0	Polymerization of high cis polybutadiene in hexane using catalyst (Euler scheme)		S5
PSS	5	5	1	0	Primary and secondary stable model		E
QUINIDIN	4	4	0	0	Population pharmacokinetics of quinidine	[102]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
RAD_TRAC	3	17	0	0	Radioactive tracer in two human body compartments		E
RAMAN	2	303	0	0	Raman intensity of anisotrope probes	[269]	S5
RAT_APP	4	11	2	2	Rational approximation with constraints	[288]	E
RAT_FIT	4	11	0	0	Fitting a rational function	[258]	E
RAT42	3	9	0	0	Pasture yield with sigmoidal growth curve (NIST study)	[395]	E
RAT43	4	15	0	0	Dry weight of onion bulbs and tops (NIST study)	[395]	E
REFLECT	6	24	0	0	Reflection model for color design		E
REFLECTY	2	125	0	0	Reflectivity evaluation of electrons		E
RELEASE	4	32	0	0	Drug release		E
RICH_GR	3	9	0	0	Richards growth model	[402]	E
ROZMAN1	4	25	0	0	Quantum defects in iodine atoms (NIST study)		E
RTD	2	26	1	0	Residence time distribution		E
SEQ_EXP	3	13	0	0	Sequential experiment	[458], [146]	E
SMOOTHING	3	170	0	0	Data smoothing		E
SPLINE	2	6	0	0	Spline test		S0
STEP_RES	3	22	0	0	Second-order equation with dead time and step response data	[534]	E
SULFATE	4	17	0	0	Compartmental analysis in humans with radioactive sulfate as tracer	[458]	E
TEMP_LEV	6	92	0	0	Temperature level of goods		E
THERMRES	3	10	0	0	Thermistor resistance, exponential data fitting		E
THURBER	7	37	0	0	Semiconductor electron mobility (NIST study)		E
TIME_ACT	2	9	0	0	Time activities		E
TP1	2	2	0	0	Rosenbrock's banana function	[222]	none
TP1_A	2	2	0	0	Rosenbrock's banana function, ill-conditioned	[222]	none
TP1_B	2	2	0	0	Rosenbrock's banana function, very ill-conditioned	[222]	none
TP13	2	2	1	0	Academic test problem without constrained qualification	[222]	none
TP14	2	2	2	1	Constrained least squares problem	[222]	none
TP2	2	2	0	0	Constrained Rosenbrock's banana function	[222]	none
TP202	2	2	0	0	Academic test problem with attractive local solution	[428]	none
TP203	2	3	0	0	Simple data fitting problem	[428]	X

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
TP205	2	3	0	0	Least squares problem with three terms	[428]	none
TP212	2	2	0	0	Least squares problem with two terms	[428]	none
TP241	3	5	0	0	Least squares problem, five polynomial functions	[428]	none
TP242	3	10	0	0	Exponential test function	[428]	X
TP244	3	10	0	0	Exponential test function	[428]	X
TP246	3	3	0	0	Least squares problem with three terms	[428]	none
TP247	3	3	0	0	Least squares problem, helical valley in x3 direction	[428]	none
TP25	3	99	0	0	Academic test problem, highly unstable	[222]	X
TP256	4	4	0	0	Least squares problem with four terms, Powell's function	[428]	none
TP260	4	7	0	0	Least squares problem with seven terms	[428]	none
TP261	4	5	0	0	Least squares problem with exponential and trigonometric terms	[428]	none
TP267	5	11	0	0	Exponential test function	[428]	X
TP269	5	4	3	3	Constrained least squares problem with four linear terms	[428]	none
TP272	6	13	0	0	Exponential test function	[428]	X
TP282	10	11	0	0	Least squares problem with quadratic terms	[428]	none
TP286	20	20	0	0	Least squares problem with quadratic terms	[428]	none
TP288	20	20	0	0	Least squares problem, 20 linear terms	[428]	none
TP303	18	20	0	0	Least squares problem with squared sum	[428]	none
TP307	2	10	0	0	Exponential data fitting	[428]	E
TP308	2	3	0	0	Least squares problem with trigonometric terms	[428]	none
TP312	2	2	0	0	Least squares problem with two quadratic terms, local solution	[428]	none
TP327	2	44	1	0	Constrained exponential data fitting	[428]	E
TP332	2	201	2	0	Cam design problem	[428]	none
TP333	3	8	0	0	Exponential data fitting	[428]	E
TP334	3	15	0	0	Exponential data fitting	[428]	E
TP350	4	6	0	0	Rational approximation	[428]	E
TP351	4	7	0	0	Rational data fitting	[428]	E
TP352	4	40	0	0	Exponential and trigonometric data fitting	[428]	E
TP354	4	4	1	0	Constrained least squares problem, four quadratic terms	[428]	none

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
TP355	4	2	3	1	Constrained least squares problem, four quadratic terms and local solutions	[428]	none
TP358	5	20	0	0	Exponential data fitting test function	[428]	E
TP370	6	87	0	0	Complex least squares problem, six variables	[428], [368]	E
TP371	9	87	0	0	Complex least squares problem, nine variables	[428], [368]	E
TP372	9	6	12	0	Least squares problem, twelve inequality constraints	[428], [368]	none
TP373	9	6	6	6	Least squares problem, six equality constraints	[428], [368]	none
TP379	11	65	0	0	Test problem of Osborne, four exponential terms	[428], [368]	E
TP394	20	40	1	1	Least squares problem with one equality constraint	[428]	none
TP43	4	1	3	0	Rosen-Suzuki test problem	[222]	none
TP46	5	4	2	2	Equality constrained academic test problem	[222]	none
TP48	5	3	2	2	Equality constrained academic test problem	[222]	none
TP57	2	44	1	0	Constrained exponential fit	[222]	E
TP6	2	1	1	1	Rosenbrock's banana function, Betts' formulation	[222]	none
TP70	4	19	1	0	Chemical equilibrium problem	[222]	E
TPGLOB	3	3	0	0	Global solution of three equations in L1-form	[428]	none
TRAJ_NET	2	21	0	0	Test function with 5 local minima	[113]	E
TRANS90	3	22	0	0	Emittor at 90 deg		S5
TRANS90X	3	69	0	0	Emittor at 90 deg, experimental data		E
TREND	6	500	1	0	Trend curve		E
TRIG_APP	2	19	0	0	Trigonometric approximation for computing axial forces		E
TUBTANK	1	19	0	0	Comparison of tank and tubular reactors steady state	[234]	S5
VAPOR	2	11	0	0	Vapor-liquid equilibrium	[128]	E
VIS_SPEC	20	200	0	0	Viscoelastic spectra		N1
VISC_ELA	10	24	0	0	Memory function of visco-elastic substances		E
VISCREG1	3	23	0	0	Viscosity regression, data set 1		E
VISCREG2	3	103	0	0	Viscosity regression, data set 2		E
VISCREG3	3	1541	0	0	Viscosity regression, all data sets		E
WAVE_X	3	80	0	0	Explicit solution of wave equation		S5
WEIBULL	2	12	0	0	Weibull distribution		S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
ZS_EQTN	2	10	0	0	Zimm-Stockmayer equation	[143]	E

10.2 Laplace Transforms

Now we assume that the data fitting function is given in form of a vector-valued Laplace transform $H(p, s, c) \in \mathbb{R}^r$ depending on the parameter vector p to be fitted, the Laplace variable s , and an optional so-called concentration parameter c . Let function $h(p, t, c)$ be a numerical approximation of the inverse Laplace transform of $H(p, s, c)$, for instance computed by the formula of Stehfest [490], separately for each component.

Proceeding now from $l = l_t l_c r$ E data (t_i, c_j, y_{ij}^k) and weights w_{ij}^k , $i = 1, \dots, l_t$, $j = 1, \dots, l_c$, and $k = 1, \dots, r$, we get the parameter estimation problem

$$p \in \mathbb{R}^n : \min_{p_l \leq p \leq p_u} \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, t_i, c_j) - y_{ij}^k))^2 .$$

General nonlinear constraints are not permitted in this case. Test problems defined by their Laplace transforms are listed in Table B.2.

Table B.2. Laplace Transforms

<i>name</i>	<i>n</i>	<i>l</i>	<i>background</i>	<i>ref</i>	<i>data</i>
CONCS	2	7	Test problem, only concentration values		S5
DIFFUS_L	1	99	Linear diffusion with constant parameters		S1
ELEC_NET	6	30	Electrical net		S5
LKIN_L	3	26	Simple linear compartment model, Laplace formulation		E
LKIN_L3	2	78	Simple linear compartment model, three initial doses		S5
PLASTER1	7	7	Pharmaceutic transdermal diffusion (plaster)	[552], [194]	E
PLASTER2	4	7	Pharmaceutic transdermal diffusion (plaster)	[552], [194]	E
PLASTER3	5	12	Plaster diffusion	[552], [194]	E
PLASTER4	2	12	Plaster diffusion	[552], [194]	E
TRAY	3	26	Marking tray hits		E

10.3 Steady State Equations

Again, it is supposed that r measurement sets of the form

$$(t_i, c_j, y_{ij}^k), \quad i = 1, \dots, l_t, \quad j = 1, \dots, l_c, \quad k = 1, \dots, r$$

are given with l_t time values, l_c concentration values, and $l = l_t l_c r$ corresponding measured E data. Moreover, we have weights w_{ij}^k . Weights can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for state variables for which E data do not exist. Thus, the subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation.

Together with an arbitrary fitting criterion $h(p, z, t, c)$, we get the parameter estimation problem

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, z(p, t_i, c_j), t_i, c_j) - y_{ij}^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0, \quad j = 1, \dots, m_e, \\ & g_j(p) \geq 0, \quad j = m_e + 1, \dots, m_r, \\ & p_l \leq p \leq p_u. \end{aligned}$$

We assume that fitting criteria $h_k(p, z, t, c)$, $k = 1, \dots, r$, state variable $z(p, t, c)$, and constraints $g_j(p)$, $j = 1, \dots, m_r$, are continuously differentiable functions subject to p .

The state variable $z(p, t, c) \in \mathbb{R}^m$ is implicitly defined by the solution z of a system

$$\begin{aligned} s_1(p, z, t, c) &= 0, \\ &\dots \\ s_m(p, z, t, c) &= 0 \end{aligned}$$

of nonlinear equations. All steady state test problems are listed in Table B.3. Since none of them possesses additional constraints, the corresponding figures m_r and m_e are omitted.

Table B.3. Steady State Equations

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>background</i>	<i>ref</i>	<i>data</i>
ABSORP	1	10	7	Adsorption with surface complexation		E
BLOOD_S	3	32	1	Blood ethanol concentration	[522]	E
CENTRI	3	11	1	Ultracentrifuge for molecular weight determination		E
CHARGE	4	9	1	Charge relulation model, zero-potential as function of pH		S1
CHEM_EQU	2	20	4	Chemical equilibrium system		E
CHEMSTAT	2	10	3	Optimal residence time of a chemostat	[128]	S5
CO2_SOL1	2	14	10	VLE model for CO2 solubility in aqueous amine solutions		E
DEFORM	3	137	0	Deformation of visco-elastic material		E
DEWPOINT	1	21	3	Dew point temperature for isobutanol and water mixture	[465], [308]	S1
DISS_ENZ	2	30	1	Inhibition of dissociative enzymes	[263]	S5
MD_EQUI	1	6	1	Monomer-dimer equilibrium	[263]	S5
MDT_EQUI	2	6	1	Monomer-dimer-tetramer equilibrium	[263]	S5
METHANE	2	12	2	Partial oxidation of methane with oxygen	[191], [308]	S5
MULT_CST	4	20	8	Four stage CSTR battery in steady-state	[534]	S5
NA_CSTR	3	7	2	Continuous-flow stirred tank reactor (steady-state, normalized)	[523], [87]	S5
PERIA	5	30	2	Pulsar problem of astronomy		S1
RECLIC19	4	13	3	Receptor-ligand binding study	[407], [145]	S1
RECLIG1	2	33	2	Saturation curve 3H-compound on receptor membrane, one receptor and one ligand	[407]	E
RECLIG10	4	10	3	Receptor-ligand binding study	[407], [145]	S5
RECLIG11	2	34	2	Displacement curve with one receptor, one ligand	[407], [145]	E
RECLIG12	4	20	3	Displacement curve with one receptor, two ligands	[407], [145]	E
RECLIG13	4	22	3	Saturation curve	[407], [145]	E
RECLIG14	7	24	4	Displacement curve of quinpirole	[407], [145]	E
RECLIG15	7	16	4	Mass equilibrium model with two receptors and two ligands	[407], [145]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>background</i>	<i>ref</i>	<i>data</i>
RECLIG16	4	20	3	Mass equilibrium model with two receptors and one ligand	[407], [145]	E
RECLIG17	2	7	2	Mass equilibrium model with one receptor and one ligand	[407], [145]	E
RECLIG18	4	14	3	Mass equilibrium model with one receptor, two ligands	[407], [145]	E
RECLIG2	3	27	3	Displacement curve of 3H-compound from one receptor, two ligands	[407], [145]	E
RECLIG3	4	10	3	Saturation curve, two receptors and one radioligand	[407], [145]	E
RECLIG4	6	75	4	Displacement curve of a 3H-compound with a substance, two receptors and ligands	[407], [145]	E
RECLIG5	4	11	3	Mass equilibrium model with one receptor and two ligands	[407], [145]	E
RECLIG6	3	12	3	Displacement curve	[407], [145]	E
RECLIG7	4	44	3	Displacement curve of 3H-compound from receptor	[407], [145]	E
RECLIG8	4	10	3	Saturation curve, two receptors and one radioligand	[407], [145]	E
RECLIG9	4	22	4	Displacement curve with cold ligand on receptor	[407], [145]	E
SING.EQU	2	7	1	Single equation		E
SS.REAC	4	30	6	Steady state reaction		S5
SULPHUR	4	90	3	Oxidation of sulphur dioxide to sulphur trioxide	[152], [308]	S1
TITRATIO	3	51	5	Potentiometric titration of N,N-dimethylaminoethylamine	[458]	S1
ULTRA1	3	6	1	Ultracentrifuge data analysis for molecular weight determination for one substance	[159]	S5
ULTRA2	3	6	1	Ultracentrifuge data analysis for molecular weight determination of two substances	[159]	S5

10.4 Ordinary Differential Equations

As before, we proceed from r data sets of the form

$$(t_i, c_j, y_{ij}^k), \quad i = 1, \dots, l_t, \quad j = 1, \dots, l_c, \quad k = 1, \dots, r, \quad ,$$

where l_t time values, l_c concentration values and $l = l_t l_c r$ corresponding measurement values are given. Furthermore, we assume that l weights w_{ij}^k are defined. However, some of the weights w_{ij}^k can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for state variables for which E data do not exist. Thus, the subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation.

The data fitting function $h(p, y(p, t, c), t, c)$ depends on a concentration parameter c and in addition on the solution $y(p, t, c)$ of a system of m coupled ordinary differential equations with initial values

$$\begin{aligned} \dot{y}_1 &= F_1(p, y, t, c) \quad , \quad y_1(0) = y_1^0(p, c) \quad , \\ &\dots \\ \dot{y}_m &= F_m(p, y, t, c) \quad , \quad y_m(0) = y_m^0(p, c) \quad . \end{aligned}$$

Without loss of generality, we assume that, as in many real life situations, the initial time is zero. The initial values of the differential equation system $y_1^0(p, c), \dots, y_m^0(p, c)$ may depend on one or more of the system parameters to be estimated, and on the concentration parameter c .

The resulting parameter estimation problem can be written in the form

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, y(p, t_i, c_j), t_i, c_j) - y_{ij}^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0 \quad , \quad j = 1, \dots, m_e \quad , \\ & g_j(p) \geq 0 \quad , \quad j = m_e + 1, \dots, m_r \quad , \\ & p_l \leq p \leq p_u \quad . \end{aligned}$$

Again we have to assume that model functions $h_k(p, y, t, c)$ and $g_j(p)$ are continuously differentiable functions of p , $k = 1, \dots, r$ and $j = 1, \dots, m_r$, and that the solution $y(p, t, c)$ is also a smooth function of p . All test problems based on ordinary differential equations are listed in Table B.4. We do not list additional information about switching points or boundary values, for example.

Table B.4. Ordinary Differential Equations

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
2BODY	2	80	4	0	0	Two-body problem	[230]	S5
2CSTR	3	80	4	0	0	Series of two CSTRs with time-delay	[366], [308]	S5
2LNK_ROB	2	40	4	0	0	Two-link planar robot without constraints	[11]	S5
2ND_ORD	3	10	2	0	0	Academic test problem, ill-behaved second order IVP	[62], [496]	S5
2ND_RATE	3	15	1	0	0	Second order rate equation under heat transfer conditions	[534]	S1
2STGCSTR	7	5	4	1	0	Time-optimal bang-bang control of two-stage CSTR	[308], [129]	none
ACTIVITY	2	9	2	0	0	Activities over time		E
ACTNITR	4	80	8	0	0	Nitrification in activated sludge process	[126]	S5
ADIABATI	2	30	2	0	0	Adiabatic complex gas-phase reaction in a PFR	[534]	S1
AEKIN	8	120	3	0	0	AE-kinetics		S1
AIRLIFT	1	87	2	0	0	Continuously stirred reactor with airlift		E
AIRY	2	38	2	0	0	Airy equation	[493]	S5
AKTIV_W2	8	128	4	0	0	Association kinetics, two-state-theory		S5
ALDRIN	2	10	1	0	0	Diffusion with chemical reaction		E
ALPHA.PI	5	52	5	0	0	Isomerization of an alpha-pinene		S5
AMENTO	5	48	12	0	0	Transport of Amentoflavone in Transwell plates		E
AMIDPRO	4	201	4	0	0	Amidproton replacement with protein folding		E
AMMONAB	3	39	3	0	0	Steady-state absorption column design	[234]	S1
AMYLASE	5	50	7	0	0	Alpha-amylase production with bacillus subtilis		S5
ANAEEMEAS	5	72	7	0	0	Anaerobic reactor activity	[126]	S1
ANHYD	2	56	3	0	0	Oxidation of o-xylene to phthalic anhydride	[234]	S1
ANTIBIO	5	20	2	0	0	Kinetics of antibiotics in liquid manure	[402]	E
APHID	3	30	2	0	0	Number of aphids		E
APHIDEX	3	126	2	0	0	Number of aphids		E
APPRX1	5	20	1	0	0	Curve fitting	[548]	X

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
APPRX2	3	4	1	0	0	Curve fitting	[548]	E
ASS_CV1	11	57	7	0	0	Association curves		E
ASS_CV2	5	31	2	0	0	Association curves		E
ASS_CV3	6	27	2	0	0	Association curves		E
ASS_CV4	5	53	2	0	0	Association curves		E
ASS_CV5	7	47	3	0	0	Association curves		E
ASS_CV6	6	23	3	0	0	Association curves		E
ASS_CV7	7	23	2	0	0	Association curves		E
ASS_KIN1	3	15	1	0	0	Association kinetics		E
ASS_KIN2	4	15	1	0	0	Association kinetics with exponential term		E
ASS_KIN3	3	37	1	0	0	Association kinetics		E
ASS_KIN4	4	11	1	0	0	Association kinetics		E
ASS_KIN5	6	16	2	0	0	Association kinetics		E
ASS_KIN6	4	16	1	0	0	Association kinetics with exponential term and slope		E
ASTRO	1	80	4	0	0	Planar motion of earth around sun (singularities)	[11]	S5
ASYMP	3	27	2	0	0	Asymptotic boundary value problem	[11]	S5
AXDISP	3	80	16	0	0	Differential extraction column with axial dispersion	[234]	S5
B_BLOCK	10	41	2	0	0	Control of beta-blocker, two compartments	[90]	E
B_BLOCK1	20	41	2	0	0	Control of beta-blocker, two compartments	[90]	E
B_BLOCK2	40	41	2	0	0	Control of beta-blocker, two compartments	[90]	E
B_LYMPHO	8	12	3	0	0	B Lymphocytes in bone marrow	[179]	S5
BACTERIA	5	60	1	0	0	Bacteria population dynamics	[325]	S5
BALL	5	1	2	5	5	Bouncing ball	[468]	X
BARN1	3	22	2	0	0	Chemical reaction, Lotka-Volterra equation	[508]	E
BARN2	5	22	2	0	0	Chemical reaction, Lotka-Volterra equation with variable initial values	[508]	E
BATCH_CT	7	1	2	0	0	Control of nonlinear batch reactor	[306]	none
BATCHD	3	19	1	0	0	Dimensionless kinetics in a batch reactor	[234]	S5
BATCOM	8	364	4	0	0	Batch reactor with complex reaction sequence	[234]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
BATEX	2	20	2	0	0	Single solute batch extraction	[234]	S5
BATFERM	4	120	3	0	0	Batch fermentation	[126]	S5
BATSEG	2	10	2	0	0	Simple reaction with segregation in a batch reactor	[234]	S5
BATSEQ	4	44	4	0	0	Complex batch reaction sequence	[234]	S5
BCBPLUS	6	164	13	0	0	Reaction between brilliant cresol blue (BCB+) and acid chlorite		E
BEAD	3	90	6	0	0	Diffusion and reaction in a spherical bead	[234]	S5
BEER	17	62	7	0	0	Beer fermentation		E
BELLMAN	3	15	1	0	0	Chemical reaction (Bellman)	[522]	E
BELUSOV	3	132	4	0	0	Oscillating chemical reaction, highly stiff (Belusov-Zhabitsky)		S5
BENZENE	2	16	2	0	0	Pyrolytic dehydrogenation of benzene to diphenyl		E
BENZHYD	2	20	2	0	0	Isothermal tubular reactor with two consecutive reactions (dehydrogenation of benzene)	[234]	S5
BLOSC	2	100	2	0	0	Chaotic bi-stable oscillator	[57], [189]	S1
BIMOLECU	3	14	1	0	0	Carcino-embryonic antigen binding, bimolecular reversible reaction	[6]	E
BIO_MOD	2	300	3	0	0	Substrate production from biomass		S5
BIO_REAC	21	441	8	0	0	Biochemical dynamic model, highly overdetermined		E
BIODEG	8	42	3	0	0	Degradation of two substrates and growth of biomass		E
BIOKAT	5	100	3	0	0	Bimolekulare catalysis, Diels-Adler-reaction	[342]	S5
BIOMASS	2	10	2	0	0	Biomass and substrate of fermentor		S5
BIOPROC	4	15	3	0	0	Recombinant microbiological process	[130]	S5
BIOREAC	6	101	2	0	0	Biochemical reaction with enzyme deactivation	[15]	S1
BITUMEN	5	27	3	0	0	Modified Denbigh reaction scheme for converting bitumen into waste	[306], [105]	S5
BLASIUS	3	202	3	1	1	Incompressible laminar flow with zero pressure gradient	[229]	S5
BLOOD	10	124	9	0	0	Blood coagulation mechanism by thrombin formation	[497]	E
BLOOD_O	3	32	1	0	0	Blood ethanol concentration	[522]	E
BLYPMPH	8	12	3	0	0	B-lymphocytes in a bone marrow	[179]	E
BLYPMPH_R	3	12	3	0	0	B-lymphocytes in a bone marrow with reduced parameter set	[179]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
BRUNHILD	6	21	3	0	0	Bolus injection of radioactive sulfate	[458]	E
BRUSSEL1	4	30	6	0	0	Multi-molecular reaction (Brusselator)	[278]	S5
BRUSSEL2	2	80	2	0	0	Multi-molecular reaction (Brusselator)	[197]	S5
BSTILL	4	180	11	50	0	Binary batch distillation column (nine floors)	[234]	S5
BSTILL_I	5	130	13	0	0	Binary batch distillation column (eleven floors)	[234]	S5
BVP	2	9	2	1	1	Boundary value problem	[11]	S5
BVP_NL	1	1	5	0	0	Nonlinear boundary value problem	[10]	none
BVP4	8	55	16	8	8	Complex 4-th order boundary value problem (normal mode decomposition of PDE)		S5
CABBAGE	8	24	3	0	0	Growth of white cabbage (roots, stem, leaves)	[402]	E
CAMY	6	568	2	0	0	Estimation of kinetic parameters of a chemical reactor		E
CAR_CTR1	21	1	2	2	2	Acceleration of a car		none
CAR_CTR2	22	1	2	3	2	Acceleration of car with possible break failure	[2]	none
CARGO	11	60	6	3	3	Transferring containers from ship to truck	[137], [502]	S1
CASC_IMP	2	10	11	0	0	Air humidity in laboratory device		S1
CASCADE1	15	9	5	1	1	Storage cascade of flow in pipes, Riccati equation	[286]	E
CASCADE2	3	9	1	0	0	Flow in pipes with one storage, Riccati-Muskingum equation	[286]	E
CASCSEQ	5	30	12	0	0	Cascade of three reactors with sequential reactions	[234]	S5
CASTOR	2	88	2	0	0	Batch decomposition of acetylated castor oil	[234]	S5
CAT_HYD	1	24	2	0	0	Catalytic hydrolysis of acetic anhydride	[523]	S5
CATALYST	10	1	7	0	0	Bifunctional catalyst blend of methylcyclopentane to benzene in a tubular reactor	[308]	none
CAV_BUBB	3	9	2	0	0	Cavitating bubble	[274]	S5
CHLCIRC	4	150	3	0	0	Electric circuit in a chaotic regime	[506]	S1
CHAIN_O1	4	40	2	0	0	First-order reversible chain reaction	[508]	S5
CHAN_FLO	3	23	4	2	2	Flow of a fluid during injection into a long channel	[118]	S1
CHANNEL	3	9	3	2	2	Flow in a channel (3rd order BVP)	[11]	S5
CHEM_OSC	10	50	5	0	0	Chemical oscillator	[223], [459]	S10
CHEM_REA	17	99	9	0	0	Chemical reaction		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
CHEMO	6	184	3	0	0	Chemostat fermentation	[126]	S5
CHSTAD	8	54	5	1	1	Chemostat with dilution rate shift up, transient system		E
CIRCLE	2	402	4	0	0	Parameterized circle equation		S5
CIRCUIT	4	60	3	0	0	Electric circuit in a chaotic regime	[506]	N1
CLOUD	2	50	2	0	0	Behavior of spherical cloud of gas under gravitation	[103]	S5
COAL1	6	13	2	0	0	Coal pyrolysis, two parallel CH4 reactions	[73], [295]	S5
COAL2	11	86	3	0	0	Coal pyrolysis, concurrent reactions including CO, CO2, CH4, H2	[419], [295]	E
COAL3	3	20	1	0	0	Coal pyrolysis, first order H2 reaction	[419], [295]	E
COAL4	6	21	2	0	0	Coal pyrolysis, two parallel CO2 reactions	[419], [295]	E
COAL5	6	23	2	0	0	Coal pyrolysis, two parallel CO reactions	[419], [295]	E
COAL6	3	21	1	0	0	Coal pyrolysis, higher order CH4 reaction	[419], [295]	E
COAL7	18	86	6	0	0	Coal pyrolysis, parallel, higher reactions including CO, CO2, CH4, H2	[419], [295]	E
COLCON	4	50	11	0	0	Extraction cascade with backmixing and control	[234]	S5
COLLISIO	2	400	8	0	0	Collision dynamics between an Argon and a Neon atom in their mutual Lennard-Jones force field	[453]	S0
COMMENSA	3	18	7	0	0	Two bacteria with opposite substrate preferences	[126]	S5
COMP_ADS	10	40	8	2	2	Competitive adsorption of two components by multilayer model		E
COMP_EXP	4	38	2	0	0	Two compartments with equal absorption and exponential elimination	[383]	S5
COMPASM	3	46	5	0	0	Competitive assimilation and commensalism	[126]	S5
COMPET	4	50	2	0	0	Competition of two species	[57], [35]	S5
COMPREAC	8	154	7	0	0	Complex reaction scheme between formaldehyde and sodium para phenol sulphonate	[234]	S5
COMPSEG	2	60	6	0	0	Complex reaction with segregation in a semi-batch reactor	[234]	S5
CON_BURG	1	22	2	10	1	Burgers' equation with state and boundary constraints	[40]	S5
CONC4	7	35	1	0	0	Chemical simulation model		E
CONC4A	7	35	3	0	0	Chemical simulation model, alternative formulation		E
CONF_ALT	6	23	2	0	0	Conformation alterations of proteins		E
CONFLO1	2	40	1	0	0	Continuous open tank flow	[234]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
CONFLO2	2	40	1	0	0	Continuous closed isothermal tank flow	[234]	S5
CONFLO3	2	40	1	0	0	Continuous closed adiabatic tank flow	[234]	S5
CONINHIB	2	70	2	0	0	Continuous culture with inhibitory substrate	[126]	S5
CONSREA	2	16	12	0	0	Consecutive reactions with variable orders from combustion chemistry (optimized design)	[13]	S5
CONSTILL	6	60	10	0	0	Continuous binary distillation column	[234]	S5
CONTCON	3	44	3	0	0	Feed rate control of inhibitory substrate in a continuous culture	[126]	S5
CONTUN	2	105	4	0	0	Controller tuning problem	[234]	S1
CONVER	6	2	6	0	0	Control of monomer conversion and number average chain length		none
COOL	2	48	9	0	0	Continuous stirred-tank cascade	[234]	S5
COOL_CRI	2	374	5	0	0	Cooling crystallization (Miller and Parsival formulation)		E
COPPER	5	151	27	0	0	Multilayer model for moisture adsorbed on copper		E
COPPER_D	4	374	21	0	0	Multilayer model for moisture adsorbed on copper with desorption		E
COPPER2	5	277	7	0	0	Multilayer model for moisture adsorbed on copper		E
CR_ELOV	4	36	2	0	0	Chemical reaction		E
CRANE	5	18	6	0	0	Optimal control of a container crane	[416]	S5
CS_REAC	2	20	4	0	0	Continuously stirred reactor	[39]	S1
CST_IORD	5	40	2	0	0	First order continuous stirred tank with cooling coil	[534]	S1
CSTOHNE	3	400	3	0	0	Competition NH-replacement without reverse reactions		E
CSTR	2	60	3	0	0	Continuous stirred-tank cascade	[234]	S5
CSTR_BM	4	76	4	0	0	CSTR, benchmark example	[39]	S5
CSTR_CTL	20	100	2	0	0	Setpoint control of continuous stirred tank reactor	[148]	none
CSTR_CTR	7	1	3	0	0	Control of continuously stirred tank reactor	[272], [308]	none
CSTR_DFT	8	594	3	3	3	Capillary of Balzers		E
CSTR_JY0	5	306	4	0	0	Filter/capillary simulation		E
CSTRCOM	3	85	5	0	0	Isothermal reactor with complex reaction	[234]	S5
CUO	6	22	4	1	1	Multilayer model for moisture adsorbed on copper surface		E
DCIMMIGR	4	63	2	0	0	Linear model with time lags and immigration		E
DCKIN	2	20	1	0	0	Kinetic reaction with badly scaled exact solution		S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
DCMDEG	4	18	20	0	0	Dichloromethane in a biofilm fluidized sand bed	[126]	S5
DEACT	3	49	3	0	0	Deactivating catalyst in a CSTR	[234]	S5
DEACTENZ	3	90	7	0	0	Reactor cascade with deactivating enzyme	[126]	S5
DECAY	3	20	3	0	0	Radioactive decay of an isotope		S5
DEGEN	1	20	2	0	0	Notorious academic example, highly degenerate	[52]	S5
DEGEN_M	1	40	2	0	0	Modified notorious academic example, highly degenerate	[52], [561]	S5
DEGRAD1	3	30	3	0	0	Microbial degradation with hydrolysis and photolysis		E
DEGRAD2	4	30	3	0	0	Microbial degradation with hydrolysis and photolysis, parallel version		E
DEHYBENZ	2	16	2	0	0	Pyrolytic dehydrogenation of benzene to diphenyl		E
DIABETES	6	20	5	0	0	Diabetes management		E
DIAUXIA	5	100	5	0	0	Diauxic growth of a microbe		S5
DIFDIST	4	36	10	0	0	Multicomponent differential distillation	[234]	S1
DIMER	4	20	2	0	0	Pharmakokinetic model with two substances and one dimer complex		E
DIODE	2	18	2	0	0	Tunnel-diode oscillator	[235]	S5
DIS_KIN1	10	168	5	0	0	Displacement kinetics of a pharmaceutical experiment, data set 1		E
DIS_KIN2	10	168	5	0	0	Displacement kinetics of a pharmaceutical experiment, data set 2		E
DISLIQU	1	144	6	0	0	Distribution of substrates in a chemical reactor, liquid phase		S5
DISORDER	3	18	2	0	0	Treating manic-depressive disorder with Lithium carbonate	[468]	S5
DISPLMNT	8	32	3	0	0	Displacement curve		E
DISRET_O	2	128	16	0	0	Non-isothermal tubular reactor with axial dispersion	[234]	S5
DISSOC	8	94	1	0	0	Dissociation kinetics		E
DISTPAR1	17	78	2	0	0	Distributed parameter system (highly unstable and overdetermined)		S0
DISTPAR2	18	78	2	0	0	Distributed ODE parameter system, data from DISTPAR1		E
DLA1	6	242	2	0	0	Dissociation limited association kinetics, data set 1		E
DLA2	6	242	2	0	0	Dissociation limited association kinetics, data set 2		E
DLA3	6	242	2	0	0	Dissociation limited association kinetics, data set 3		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
DLA4	6	242	2	0	0	Dissociation limited association kinetics, data set 4		E
DLA5	6	242	2	0	0	Dissociation limited association kinetics, data set 5		E
DLA6	6	242	2	0	0	Dissociation limited association kinetics, data set 6		E
DLA7	6	242	2	0	0	Dissociation limited association kinetics, data set 7		E
DLA8	6	242	2	0	0	Dissociation limited association kinetics, data set 8		E
DMDS	8	66	4	0	0	Catalytic conversion of dimethyldisulfide		E
DMDS_A	8	88	4	0	0	Transient catalytic conversion of dimethyldisulfide		E
DRUG_SCH	3	32	3	0	0	Optimal drug scheduling for cancer chemotherapy	[77]	S5
DRUGDIS1	2	3	2	0	0	Time-optimal drug displacement, warfarin and phenylbutazone, one jump	[308], [326]	none
DRUGDIS2	4	3	2	2	0	Time-optimal drug displacement, warfarin and phenylbutazone, three jumps	[308], [326]	none
DRY_FRI1	1	40	4	0	0	Two-mass oscillator with dry friction between bodies (implicit switching)	[132]	S0
DRY_FRI2	3	40	4	1	0	Two-mass oscillator with dry friction between bodies, two variable switching times	[132]	S0
DRY_FRI3	5	40	4	3	0	Two-mass oscillator with dry friction between bodies, four variable switching times	[132]	S0
DRYER	5	63	4	0	0	Non-isothermic shaft dryer		S5
DUAL	3	48	3	0	0	Dual substrate limitation	[126]	S5
DUCT	3	10	1	0	0	Duct design problem (boundary value problem)	[56]	S5
DYNAMO	2	120	3	0	0	Chaotic behaviour of coupled dynamos	[57], [35]	S1
EAR2MAR1	10	1	3	3	3	Optimal control of earth-to-mars transfer	[61]	none
EAR2MAR2	51	1	3	3	3	Optimal control of earth-to-mars transfer	[61]	none
ENTERO	4	27	4	0	0	Linear pharmaco-kinetic model with lag-time		E
ENZCON	3	51	3	0	0	Continuous enzymatic reactor	[126]	S5
ENZSPLIT	3	10	2	1	1	Diffusion and reaction: split boundary solution	[234]	S1
ENZTUBE	2	10	1	0	0	Tubular enzyme reactor	[126]	S5
ENZYM	6	28	2	0	0	Enzyme effusion problem	[522]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
EQBACK	3	50	10	0	0	Multistage extractor with backmixing	[234]	S5
EQEX	2	15	2	0	0	Simple equilibrium stage extractor	[234]	S5
EQMULTI	3	50	10	0	0	Continuous equilibrium multistage extraction	[234]	S5
ESTER_GA	3	14	1	0	0	Esterification		E
ETHANOL	7	100	4	0	0	Ethanol fed-batch fermentation by <i>S. cerevisiae</i>	[156]	S5
ETHFERM	8	69	7	0	0	Ethanol fed batch diauxic fermentation	[126]	S5
ETHYL	4	-1	3	0	0	Ethylene hydrogenation model (data sets 1 and 2)		E
ETHYL1	4	1526	3	0	0	Ethylene hydrogenation model (data set 1)		E
ETHYL2	4	1421	3	0	0	Ethylene hydrogenation model (data set 2)		E
EX_BREAK	5	26	2	0	0	Linear compartment model with application of 2nd dose		S0
EXO_REAC	6	157	4	0	0	Exothermic reaction with lag time		E
EXOTHERM	2	100	2	0	0	Exothermic n-th order reaction in closed vessel (normalized)	[523]	S5
EXP_INC	2	60	3	0	0	Exponentially increasing solutions	[561], [10]	N1
EXP_SIN	2	7	1	0	0	Exponential-sinus function	[468]	X
EXP_SOL	2	23	2	0	0	Exponential solution	[493]	S5
FAST	4	28	2	0	0	Test problem, fast steady-state	[493], [267]	S5
FBR	3	45	8	0	0	Fluidized bed recycle reactor	[126]	S5
FC_EVAP	3	33	3	0	0	Forced-circulation evaporator	[357]	S5
FED_BAT	4	10	2	0	0	Optimal feeding strategy for monod-type models by fed-batch experiments	[344]	S5
FED_BATE	4	10	2	0	0	Optimal feeding strategy for monod-type models by fed-batch experiments, time-dependent feed	[344]	S5
FED10	4	80	8	0	0	Fed-batch reactor for protein production by recombinant bacteria	[276], [307]	S5
FEDBAT	4	180	4	0	0	Fed batch fermentation	[126]	S5
FEDBATCH	25	192	12	0	0	Fed batch fermentation process of streptomyces tendae		E
FERMENT	3	56	5	0	0	Batch fermentation		S5
FERMNT	5	126	9	0	0	Fermentation model with jump in input function		S5
FERMPROC	4	150	3	0	0	Fermentation process in bioreactor with jump in dilution rate		S5
FERMTEMP	4	100	5	0	0	Temperature control of fermentation	[126]	S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
FIN	2	8	2	0	0	Temperature in a long fin	[32]	E
FISH_POP	8	30	3	0	0	Fish population of lake Baikal		E
FLUID_CL	2	10	2	1	1	Fluid with immersed cooling coil (BVP)	[534]	S5
FLUOR	7	11	6	0	0	Fast fluorescence rate of photosynthesis	[14]	E
FLUORES	8	38	39	0	0	Fluorescence induction problem	[494]	E
FLUORESC	8	152	39	0	0	Fluorescence induction problem	[494]	E
FOLDING1	7	69	4	0	0	Unfolding and refolding of ribonuclease T1	[364], [328]	E
FOLDING2	6	72	4	0	0	Unfolding and refolding of ribonuclease T1	[364], [328]	E
FOLDING3	5	42	4	0	0	Unfolding and refolding of ribonuclease T1	[364], [328]	E
FOLDING4	4	38	3	0	0	Unfolding and refolding of ribonuclease T1	[364], [328]	E
FOLDING5	5	38	5	0	0	Unfolding and refolding of ribonuclease T1	[364], [328]	E
FOREST	5	40	2	0	0	Growth of forest	[57]	S5
FRACTAK	7	24	2	0	0	On-off-kinetics of fractakine binding		E
FUNGI	13	11	1	0	0	Spread of fungi in the root systems of growing plants	[411], [67]	E
FUNGLI	3	11	1	0	0	Spread of fungi in the root systems of growing plants	[411], [67]	E
FUP_OSCI	2	200	40	0	0	Series of masses coupled by springs (Fermi-Ulam-Pasta oscillator)	[91], [147]	S5
GAS_ABS1	2	100	20	0	0	N-plate gas absorber with constant inlet feed stream, 20 plates	[308]	S5
GAS_ABS2	2	100	200	0	0	N-plate gas absorber with constant inlet feed stream, 200 plates	[308]	S5
GAS_OIL	3	40	2	0	0	Catalytic cracking of gas oil	[508]	S5
GASCLOUD	2	26	2	0	0	Thermal behavior of a spherical cloud of gas	[493], [471]	S0
GASLIQ1	2	20	6	0	0	Gas-liquid mixing and mass transfer in a stirred tank	[234]	S1
GASLIQ2	3	30	6	0	0	Gas-liquid mixing and mass transfer in a stirred tank	[234]	S5
GIBBFUJI	14	948	8	0	0	Modelling gibberella fujikuroi growth and GA3 production in solid state fermentation	[167]	E
GLIDER	4	72	4	1	0	Flight of glider with upwind	[530]	S1
GLOB_CO2	5	161	7	0	0	Global CO2 model, exchange of energy, water, and carbon between continents and atmosphere	[461]	S5
GLUCOSE	9	40	3	0	0	Glucose reaction	[386]	S5
GLUCOSE1	4	27	2	0	0	Minimal model for glucose and insulin kinetics	[415]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
GLUCOSE2	8	54	3	0	0	Minimal model for glucose and insulin kinetics	[415]	E
GOLF	2	24	6	0	0	Flight of a golf ball	[259]	S5
GREASE	3	13	2	0	0	Grease film of a fluid under high pressure	[259]	E
GROWTH_H	2	50	1	0	0	Logistic growth with stock dependent harvest	[57]	S5
GYROS	3	80	7	0	0	Idealized gyroscope in terms of quaternions (integral invariant)	[132]	S0
GYROSCOP	2	48	3	0	0	Heavy symmetric gyroscope	[259]	S5
HAMILTON	3	1	6	2	2	Hamiltonian system, two-point boundary system	[245]	E
HEATEX	3	200	24	0	0	Dynamics of a shell-and-tube heat exchanger	[234]	S1
HIGH_ORD	1	1	7	0	0	Ordinary differential equation of order 7		X
HIRES	11	32	8	0	0	Growth and differentiation of plant tissue at high levels of irradiance by light	[199]	S5
HIRES_PA	5	32	8	0	0	Growth and differentiation of plant tissue, after priority analysis	[199]	S5
HIRUDIN	9	39	5	0	0	Hirudin binding to thrombin and to a chemical mutant of thrombin	[264]	S5
HIV	4	40	4	0	0	HIV-AIDS epidemic evolution	[463]	S5
HMT	2	42	2	0	0	Semi-batch manufacture of hexamethylenetriamine	[234]	S5
HOLD	6	8	1	0	0	Ligament material properties with nonlinear springs and dashpots		E
HOLDUP	3	48	7	0	0	Transient holdup profiles in an agitated extractor	[234]	S1
HOLE	3	38	1	0	0	Academic test example with hole	[468]	S5
HOMPOLY	2	21	3	0	0	Homogeneous free-radical polymerization	[234]	S1
HYDROL	2	6	2	0	0	Batch reactor hydrolysis of acetic anhydride	[234]	S5
HYDROXY	2	36	1	0	0	CSTR approximation to plug flow reactor		E
IDENT1	4	31	2	0	0	Structurally globally identifiable model	[538]	S0
IDENT2	4	11	1	0	0	Gas production by metal dissolution of Volmer-Heyrovski	[538]	S0
IMPULSE	3	20	2	0	0	Impulse of nerve potential	[464]	S5
INC_STIF	2	14	2	0	0	Class of test problems with increasing stiffness	[249]	S5
INHIB	3	39	4	0	0	Gas and liquid oxygen dynamics in a continuous fermenter	[126]	S5
INTERLEU	16	63	28	0	0	Interleukin-13 binding kinetics	[265]	E
IONTRAN2	2	297	1	0	0	Ion transport through membrane, logistic differential equation		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
IRB6400	9	6	6	0	0	Optimal control model for the industrial robot IRB6400	[216]	none
ISO_2PHA	3	40	4	0	0	Van-de-Vusse reaction in isotherm, ideally mixed CSTR with two phases		S5
ISO_BAT	4	15	4	0	0	Ideal isothermal batch reactor	[128]	S5
ISOMER	5	40	5	0	0	Thermal isomerization of alpha-pinene to dipentene	[508], [58], [458]	E
ISOTOP1	15	108	9	0	0	Isotope dilution with nine compartments		E
ISOTOP2	28	108	9	7	7	Isotope dilution with nine compartments		E
JFIT	7	24	1	0	0	Chemical reaction		E
JUXTA	4	10	4	0	0	Site juxtaposition model		E
KATALY1	13	49	9	0	0	Test reaction for catalysts		E
KATALY2	19	192	12	0	0	Test reaction for catalysts		E
KEPLER	2	48	4	0	0	Modified Kepler problem	[11], [417], [198]	S5
KIDNEY	4	200	5	0	0	Class of stiff test problems	[455]	S5
KIN_PRO	7	130	10	0	0	Kinetic chemical process		E
KINMOD	3	102	1	0	0	Reaction kinetics		S5
KLADYN	3	80	4	0	0	Dynamic model for KLa	[234]	S5
KNEE	1	9	1	0	0	Knee problem	[100]	S5
LANDING	3	202	6	0	0	Emergency landing of a hypersonic flight system		S5
LASER	3	36	6	0	0	Amplify electro-magnetic radiation by stimulated emission	[38]	S5
LEG_POL	2	18	2	0	0	Legendre polynomial of order 2	[493]	X
LEPS	3	600	6	0	0	LEPS-contour of molecule D-C-H	[453]	S5
LIN_SYS	1	210	15	0	0	System of linear ODEs	[421]	S5
LINEWEAV	2	15	1	0	0	Lineweaver-Burk plot	[126]	S5
LISA	5	6	7	0	0	Low thrust orbital transfer of a LISA spacecraft	[524]	E
LKIN	3	26	2	0	0	Simple linear compartment model		E
LKIN_A1	4	26	2	0	0	Simple linear compartment model, one parameter overdetermined		E
LKIN_A2	3	26	2	0	0	Simple linear compartment model, two parameters overdetermined		E
LKIN_A3	6	26	2	0	0	Simple linear compartment model, three parameters overdetermined		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
LKIN_A4	3	26	2	0	0	Simple linear compartment model, one nearly redundant parameter		E
LKIN_LA	3	34	2	0	0	Simple linear compartment model with variable lag time		S5
LKIN_NUM	3	26	8	0	0	Simple linear compartment model, explicit derivatives		E
LKIN_O3	2	78	2	0	0	Simple linear compartment model with three doses		S5
LKIN_RE	3	26	2	21	0	Simple linear compartment model, dynamic constraints		E
LKIN_S	3	26	8	0	0	Simple linear compartment model with sensitivity equations		E
LKIN_T	3	26	2	0	0	Simple linear compartment model (ODE), approximation error		E
LOC_EQUI	4	10	2	0	0	Location and continuation of equilibria		S1
LOG_GROW	2	50	1	0	0	Logistic growth with constant harvest	[57], [302]	S5
LORENZ	6	16	3	0	0	Lorenz equation	[297]	S5
LORENZ_S	3	240	3	0	0	Lorenz equation, highly oscillating	[297]	S5
LOT_VOL1	3	200	2	0	0	Lotka-Volterra differential equation	[225]	S5
LOT_VOL2	4	20	2	0	0	Lotka-Volterra differential equation	[508]	S5
LYMPHO	10	90	12	0	0	Immune lymphocyte circulation	[338]	S5
MARINE	16	160	8	0	0	Marine population	[118]	E
MCSTILL	3	100	20	0	0	Continuous multicomponent distillation column	[234]	S1
MCSTILLX	8	110	20	0	0	Continuous multicomponent distillation column, design of input feed	[234]	ED
MECH_SYS	6	230	4	0	0	Mechanical oscillating system with elasticity, slack, and damping		E
MEMINH	3	70	3	0	0	Cell retention membrane reactor	[126]	S5
MEMSEP	2	36	6	0	0	Gas separation by membrane permeation	[234]	S5
MENDES	36	1680	8	0	0	Biochemical dynamic model		S5
MET_SURF	6	46	2	0	0	Metalloid surface		E
METHAN	6	48	3	0	0	Conversion of methanol to various hydrocarbons	[318]	E
METHYL	2	30	2	0	0	Thermal explosion of methyl nitrate (normalized)	[523]	S5
MICGROW	2	20	3	0	0	Fed-batch bioreactor with one growing biomass	[22]	S5
MICGROWX	4	20	3	8	0	Fed-batch bioreactor with one growing biomass, experimental design	[22]	ED

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
MICGROWY	23	20	3	37	0	Fed-batch bioreactor with one growing biomass, experimental design with input function	[22]	ED
MICGROWZ	23	5	3	37	0	Fed-batch bioreactor, experimental design with input function and weights	[22]	ED
MILK1	5	45	3	0	0	Mastitis with diapedesis of neutrophil, three equations		E
MILK2	4	45	4	0	0	Mastitis with diapedesis of neutrophil, four equations		E
MINWORLD	4	117	3	0	0	Mini-world with population, consumption, and environmental pollution	[57], [329]	S5
MIX_RAT1	3	7	1	0	0	Mixed rate model, chemical reaction		E
MIX_RAT2	3	11	1	0	0	Mixed rate model, chemical reaction		E
MIX_RAT3	6	11	1	0	0	Mixed rate model, chemical reaction (cubic fit for Qd0)		E
MIXPOP	2	180	3	0	0	Predator-prey population dynamics	[126]	S1
MM_META1	2	80	4	0	0	Metabolic process in urine and plasma, Michaelis-Menten kinetics	[245]	S0
MM_META2	2	80	4	0	0	Metabolic process in urine and plasma, Michaelis-Menten kinetics	[245]	S0
MMKINET	4	22	3	0	0	Kinetics of enzyme action	[126]	S5
MN_CTRL	7	10	1	1	1	Minimum-norm optimal control problem		none
MOISTURE	4	6	3	0	0	Moisture of granulates		E
MOON	1	10	1	0	0	One-dimensional earth-moon-spaceship problem	[350]	S5
MOT_TSP	9	40	4	3	3	Optimal control of motorized traveling salesman problem	[416]	S1
MOTION	3	62	4	0	0	Motion of a car in a arena	[11]	S5
MUBATCH	4	30	8	0	0	Multicomponent batch distillation	[234]	S5
MULTILAY	13	23	7	1	1	Multilayer model for adsorption/desorption of molecules onto a metallic surface		E
MUSCLE	5	320	3	0	0	Ca2+ release in skeletal muscle cells	[225]	S5
MYLESTR	10	32	4	0	0	Methyl ester hydrogenation	[34], [308]	E
NC_RHS	9	56	3	0	0	Non-continuous right-hand side		E
NITRIF	4	45	4	0	0	Batch nitrification with oxygen transfer	[126]	S5
NITRO	3	60	2	0	0	Conversion of nitrobenzene to aniline	[234]	S1
NITROGEN	3	14	1	0	0	Reversible homogeneous gas-phase reaction of nitrogen oxide	[508]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
NL_CSTR	10	160	8	0	0	Nonlinear isothermal CSTR for photochemical reaction	[55], [18]	S5
NL_ODE	3	20	3	0	0	Nonlinear ODE		S5
NLIN_VI1	29	248	10	0	0	Nonlinear-viscoelastic material law in frequency domain		E
NLIN_VI2	29	500	10	9	0	Nonlinear-viscoelastic material law in frequency domain with constraints		E
NON_DIFF	3	26	2	0	0	Simple linear compartment model, non-continuous RHS		S0
NON_ISO	2	20	2	0	0	Non-isothermal reactor with time-dependent reactant and temperature		S5
NON_KIN	6	30	2	0	0	Nonlinear pharmacokinetic reaction		S5
NOSTR	2	74	3	0	0	Non-ideal stirred-tank reactor	[234]	S5
NTA1	9	38	3	0	0	<i>C. heintzii</i> grown on glucose switched to nitrilo-triacetic acid		E
NTA2	10	38	3	0	0	<i>C. heintzii</i> grown on glucose switched to nitrilo-triacetic acid		E
NUTRITI	4	40	4	0	0	Nutritive cycle with two competing plant populations	[57]	S1
OBSERV1	2	11	2	0	0	Linear observer in normal form		S5
OBSERV2	2	48	4	0	0	Linear observer		S5
OBST_CTL	29	58	3	29	0	Obstacle problem (optimal control)	[454]	X
OC_EX3	6	30	2	0	0	Optimal control test problem	[137], [349]	none
OC_EX4	1	50	2	0	0	Optimal control test problem (bang-bang solution)	[137], [349]	none
OEKOSYS	8	60	3	0	0	Ecological system with two trophic layers	[294]	S5
OIL	4	40	6	0	0	Oil shale pyrolysis	[546], [308]	S1
OLIGO	4	84	4	0	0	Oligosaccharide production in enzymatic lactose hydrolysis	[126]	S5
ON_OFF1	7	8	2	1	0	On-off kinetics with two lag times		E
ON_OFF2	8	31	2	0	0	On-off kinetics		E
ON_OFF3	10	35	3	0	0	On-off kinetics binding atropin-chase		E
ON_OFF4	10	67	4	0	0	On-off kinetics binding atropin-chase		E
ON_OFF5	6	35	2	0	0	On-off kinetics binding atropin-chase		E
ON_OFF6	10	27	3	0	0	On-off kinetics binding atropin-chase		E
ON_OFF7	8	30	2	0	0	On-off kinetics binding atropin-chase		E
OPT_CON	50	1	5	0	0	Yeo's optimal control problem	[309]	none

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
OPT_CONT	4	29	3	2	0	Optimal control problem with 2nd order state constraints	[530], [60]	S5
OPT_CTRL	21	19	5	4	4	Optimal control with four final states	[89], [306]	none
OPT_KIN	6	60	2	2	0	Optimal adoption of initial infusion and doses at given therapeutic level		E
ORB_MOTN	3	40	4	0	0	Simple orbit motion	[421]	S5
ORBIT	9	3	3	0	0	Minimum time orbit transfer (optimal control)	[137]	none
OREGO	3	24	3	0	0	Belusov-Zhabotinskii reaction (oregonator)	[199], [141]	E
OSC_REAC	3	25	3	0	0	Chemical oscillation		S5
OSC_TRAN	5	86	2	0	0	Oscillating system with transient influences		E
OSC2INTM	3	364	4	0	0	Oscillation of the concentration of two intermediates	[183]	S5
OSCIL	3	150	3	0	0	Oscillating tank reactor behavior	[234]	S5
OSCILL	5	115	6	0	0	Oscillations in a heterogeneous catalytic reaction		S1
OXDYN	2	29	3	0	0	Oxygen uptake and aeration dynamics	[126]	S5
OXENZ	2	24	3	0	0	Aeration of a tank reactor for enzymatic oxidation	[126]	S1
OXIDAT	3	40	3	0	0	Oxidation reaction in an aerated tank	[234]	S5
OZONE	2	30	2	0	0	Ozon kinetics in atmosphere	[468]	S5
PAR_SIZE	4	28	30	0	0	Population balance including coalescence and breakup rates	[487]	E
PARTICLE	2	2	4	2	2	Particle diffusion and reaction (2nd order BVP)	[11]	X
PCB	17	90	9	0	0	Kinetic analysis of catalytic hydrogenization of PCB		E
PEAKS	3	102	3	0	0	Stiff ODE with sharp peaks	[180]	S5
PECAN	4	9	2	0	0	Population size of pecan aphids		E
PEND_ELA	1	20	4	0	0	Elastic pendulum	[299], [132]	S0
PENDU_I0	2	80	5	0	0	Plain pendulum, index-0-formulation		S1
PENICILL	4	40	4	0	0	Fed-batch fermentor for biosynthesis of penicillin	[307], [99]	S5
PESTICID	4	20	4	0	0	Pesticide degradation with explicit microbial population dynamics	[401], [371]	E
PHA_DYN1	9	35	3	0	0	Pharmaco-dynamic model with variable individual initial lag-times		E
PHA_DYN2	19	35	3	0	0	Pharmaco-dynamic model with variable initial values, initial time		E
					0			

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
PHA_DYN3	4	35	3	0	0	Pharmaco-dynamic model with one fixed initial time for all experiments		E
PHA_DYN4	5	35	3	0	0	Pharmaco-dynamic model with one variable initial lag-time		E
PHA_DYN5	7	61	3	0	0	Pharmaco-dynamic reaction with lag time, I0 = 0.01		E
PHA_DYN6	7	61	3	0	0	Pharmaco-dynamic reaction with lag time, I0 = 0.1		E
PHA_DYN7	7	61	3	0	0	Pharmaco-dynamic reaction with lag time, I0 = 1		E
PHA_KIN1	6	11	3	3	2	Linear pharmaco-kinetic model with bolus administration	[217]	S5
PHA_KIN2	9	52	7	1	1	Linear pharmaco-kinetic model with 3 segment absorptions, single dose	[217]	S5
PHA_REAC	5	180	2	0	0	Pharmaco-dynamic reaction		E
PHARM_AP	2	10	1	0	0	Pharmaceutical application in spherical coordinates		S1
PHARMA	9	52	5	0	0	Linear compartmental pharmacological model		E
PHB	4	60	3	0	0	Structured model for PHB production	[126]	S5
PHOS_TRA	26	58	8	0	0	Reversible reactions of phosphotransfer system		E
PHOSPH_D	2	75	3	0	0	Chemical reaction, phosphorescence		S5
PHOTO	11	38	39	0	0	Photosynthesis process	[409], [494]	E
PHOTO_PR	2	24	1	0	0	Daily photoproduction of plants	[57]	S5
PHOTO_S	11	31	39	0	0	Photosynthesis process	[409], [494]	S0
PHOTOCON	7	705	3	0	0	Fast transient photoconductivity		E
PLANT_GR	8	36	2	0	0	Plant growth (reset of initial values)		S5
PLASMID	2	200	5	0	0	Stability of recombinant microorganisms	[126]	S5
PLATINUM	6	120	3	0	0	CO oxidation on platinum	[70]	S1
PLUG_FLO	3	20	2	0	0	Plug-flow tubular reactor	[77]	S1
POLLUTNT	5	46	2	0	0	Treatment of pollutant		E
POLY1	14	17	5	0	0	Polymerization		E
POLY2	8	13	4	0	0	Polymerization		E
POLYBU	5	68	5	0	0	Polymerization of high cis polybutadiene in hexane using catalyst		S5
POLYMER	3	7	2	0	0	Polymerization		E
POPUL	10	83	10	1	1	Population counts		S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
PROTEIN	4	60	5	0	0	Production of secreted heterologous protein in fed-batch reactor by a yeast strain	[374], [18]	S5
PROTOZOA	3	21	1	0	0	Logistic growth model of protozoa		S5
PYRIDIN	10	66	7	0	0	Denitrogenization of pyridin	[51]	E
PYROLYS	2	101	6	0	0	Pyrolysis of chemically treated biomass, thermogravimetry analysis of a sample		S5
RABBIT	3	80	2	0	0	Rabbits eat grass on an island		S5
RAMP	5	62	1	0	0	Ligament material properties with nonlinear springs and dashpots		E
RATE_MOD	6	36	3	0	0	Catalytic hydrodesulfurization of sulfur molecules (DBT)		E
RATSOL1	3	20	2	0	0	Existence of rational solution	[175]	S5
RATSOL2	2	20	2	0	0	Existence of rational solution	[175]	S5
RC_CAT	3	83	1	0	0	Rate constant of catalyst C4H10O and water		E
RD_CSTR	5	1194	6	1	0	Release distribution in liquid phase of CSTR with input control		E
RE_ENTRY	7	6	6	0	0	Apollo re-entry problem	[496]	E
REAC	3	40	10	0	0	Chemical reaction		S5
REAC_CTR	7	1	2	20	0	Control of first-order reversible chemical reaction with dynamic constraints	[255], [306]	none
REACMECH	5	64	5	0	0	Reaction mechanism with stiff differential equations	[453]	S5
REACTION	6	12	5	0	0	Chemical reaction		E
REFRIG	2	10	2	0	0	Auto-refrigerated reactor	[234]	S5
REG_RES	4	100	2	0	0	Dynamics of a population depending on a regenerative resource	[57], [176]	S5
RELAY	2	50	2	0	0	Simple discontinuous model with a relay	[469]	S1
REPFED	5	40	3	0	0	Repeated fed batch culture	[126]	S5
REPLCUL	5	14	2	0	0	Repeated medium replacement culture	[126]	S5
RES_TIME	1	1	4	0	0	Optimal residence time for maximum yield in an ideal isothermal batch reactor	[128]	E
RESPIRA	6	101	2	0	0	Viscoelastic model of respiratory mechanics	[504]	N1
REVTEMP	2	48	4	0	0	Reversible reaction with variable heat capacities	[234]	S5
REXT	2	82	5	0	0	Reaction with integrated extraction of inhibitory product	[234]	S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
ROB_ARM	3	20	4	0	0	Robot arm with two links	[363]	E
ROB_CTRL	5	3	4	5	2	Time-optimal control of two-link robotic arm	[544], [308]	none
ROBERT	3	33	3	0	0	Robertson's differential equation for reaction rates	[405]	S5
ROBOT	8	102	6	0	0	Move equations of a robot (Manutec r3)	[370]	S5
ROBOT_2	17	6	4	4	0	Optimal control of robot with two arms	[532]	none
RODC	2	10	2	0	0	Radiation from metal rod	[234]	S5
ROESSLER	3	21	3	0	0	Roessler differential equation		S5
RT_PULSE	7	1	3	0	0	Rectangular pulse in right-hand side	[505], [308]	none
RUN	1	24	4	0	0	Relief on a runaway polymerization reaction	[234]	S1
SAT_EXP	6	33	3	0	0	Saturation experiment in pharmaceuticals		E
SBR_COOL	5	40	7	0	0	Optimal control of semi-batch reactor with cooling	[2]	none
SCP	2	1	5	4	0	Singular control problem	[531]	none
SE	2	29	1	0	0	Single chemical reaction		E
SEIR	5	123	6	1	1	SEIR epidemic model applied to SARS		S5
SEMPAR	2	15	5	0	0	Parallel reactions in a semi-continuous reactor	[234]	S5
SEMISEG	3	30	3	0	0	Simple reaction with segregation in a semi-batch reactor	[234]	S5
SEMISEQ	2	30	5	0	0	Sequential reactions in a semi-continuous reactor	[234]	S5
SENS	6	51	3	0	0	Stiff academic test problem for testing sensitivity analysis	[247]	S5
SHAKER	5	42	2	0	0	Shaker table driven by DC motor and transmission	[283]	S5
SHARP1	1	48	2	0	0	Sharp fronts	[81]	S1
SHARP2	1	40	2	0	0	Sharp fronts	[81]	S5
SHEEP	6	138	9	0	0	Transport of radiocaesium in sheep	[160]	S5
SHELL	6	44	9	0	0	Chemical reaction of aromates	[51]	E
SIMP_ECO	4	20	2	0	0	Simple ecological system	[453]	S5
SIMP_POP	1	15	1	0	0	Simple population model	[404]	S5
SKIN_D	5	25	30	0	0	Transdermal diffusion (discretized PDE)	[54]	E
SLIP	12	51	4	6	0	Oscillating system with transient influences and slip		E
SLUDGE	12	194	7	0	0	Thermal decomposition of wastewater sludge by thermogravimetry		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
SMALLEX	2	22	2	0	0	Small example with analytical solution		X
SOOT	2	40	2	0	0	Sooting methane and ethylene flames for LII and Raman measurements	[450]	S5
SPBEDRTD	2	90	9	0	0	Spouted bed reactor mixing model	[234]	S5
SPRING	2	101	2	0	0	Linear spring (unforced harmonic oscillator)	[250]	S0
SS.TUBE	2	6	2	1	1	Diffusion-convection in a tube, steady-state		E
SSHEATEX	2	75	3	0	0	Steady-state, two-pass heat exchanger	[234]	S05
STABCSTR	2	96	2	0	0	Stabilizing nonlinear continuous stirred tank		S5
STABIL	1	21	2	0	0	Stability of chemical reactors with disturbances	[234]	S1
STAGED	4	30	6	0	0	Two-stage culture with product inhibition	[126]	S5
STAR	3	10	4	0	0	Motion of a star within the potential of a cylindrical galaxy		E
STARS	2	183	60	0	0	Gravitational 10-body problem	[275]	S0
STAT5	3	26	4	0	0	Nucleocytoplasmic cycling as a remote censor by phosphorylation of STAT5	[499]	S5
STIFF	5	81	9	0	0	Stiff differential equation	[327]	S5
STIFF.DE	3	30	3	0	0	Stiff ODE	[467]	S5
STIFF.EQ	5	24	2	0	0	Stiff ODE	[549]	S5
STIFF1	3	27	3	0	0	Stiff test problem	[493]	X
STIFF2	4	36	2	0	0	Stiff test problem	[493]	X
SUB_ACCU	4	21	3	0	0	Substrate accumulation with two-phase aerosols		E
SUBTILIS	2	90	7	0	0	Growth and product formation of <i>B. subtilis</i> in batch or continuous culture on molasses	[126]	S5
SULFUR	9	31	5	1	0	Radioactive sulfur		E
SUPEROX1	3	164	1	0	0	Dismutation of superoxid ion to H2O and O2		E
SUPEROX2	5	198	1	0	0	Catalyzed reaction of superoxid ion		E
TAB_DIS1	7	94	1	0	0	Immediate release of solid dosage forms, extended model	[298]	E
TAB_DIS2	4	93	1	0	0	Immediate release of solid dosage forms, discrete distribution	[452]	E
TAB_DIS3	5	98	1	0	0	Immediate release of solid dosage forms, lognormal distribution	[298]	E
TAB_DIS4	5	98	1	0	0	Immediate release of solid dosage forms, normal distribution	[298], [452]	E
TAB_DIS5	4	98	1	0	0	Immediate release of solid dosage forms, chi-square distribution	[298], [452]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
TAB_DIS6	4	98	1	0	0	Immediate release of solid dosage forms, Weibull distribution	[298], [452]	E
TAB_DIS7	5	98	1	1	0	Immediate release of solid dosage forms, Bateman distribution	[298], [452]	E
TAB_DIS8	23	245	5	0	0	Immediate release of solid dosage forms, all distributions	[298], [452]	E
TANK	2	18	1	0	0	Single tank with n-th order reaction	[234]	S1
TANKBLD	2	9	2	0	0	Liquid stream blending	[234]	S5
TANKDIS	1	11	1	0	0	Ladle discharge problem	[234]	S5
TANKHYD	3	40	3	0	0	Interacting tank reservoirs	[234]	S1
TEFLON1	7	129	2	0	0	Viscous-elastic material	[253]	E
TEFLON2	3	451	2	0	0	Viscous-elastic material	[253]	E
TEMPCONT	2	29	2	0	0	Feedback control of a water heater	[126]	S5
TEST_0DE	2	10	1	0	0	Nonlinear ODE with rhs $((a*\log(y)+b)/(t+2)-5)*y$		S5
THERM	2	36	2	0	0	Thermal stability of a CSTR	[234]	S5
TOLUENE	7	39	3	0	0	Toluene hydrogenation by catalytic reaction	[34], [308]	E
TOPTCSTR	4	240	4	0	0	Time suboptimal control of two-stage continuous stirred tank		S5
TRANSUS1	5	72	6	0	0	Indinavir in Caco-2 without Liposomen and with Vinblastin		E
TRANSUS2	5	66	6	0	0	Indinavir in Caco-2 without Liposomen		E
TREES	2	16	1	0	0	Growth of trees		E
TRICHO	4	44	2	0	0	Unstructured growth of trichosporon cutaneum	[19]	S5
TRILEPT	6	157	4	0	0	Estimation of kinetic parameters of a chemical reactor		E
TUBDYN	3	200	8	0	0	Dynamic tubular reactor, eight tanks in series with nth-order reactions	[234]	S5
TUBED	2	19	1	0	0	Axial concentration profile in a tubular reactor	[234]	S5
TUBEMIX	3	54	6	0	0	Non-ideal tube-tank mixing model	[234]	S5
TURBCON	3	90	4	0	0	Turbidostat response	[126]	S5
TWOEX	3	128	4	0	0	Complex two-solute batch extraction with interchanging equilibria	[234]	S5
TWOONE	2	20	3	0	0	Competition between organisms	[126]	S5
TWOSTAGE	2	40	4	0	0	Two-stage chemostat with additional stream	[126]	S5
TWOTANK	3	80	2	0	0	Two tank level control	[234]	S1
UPTAKE	9	51	5	0	0	Compartment model for uptake, metabolic conversion and excretion of a drug		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m</i>	<i>m_r</i>	<i>m_e</i>	<i>background</i>	<i>ref</i>	<i>data</i>
US_CSTR	3	72	2	0	0	Unsteady state of a complex reaction in a CSTR	[534]	S5
VAR_META	6	504	8	0	0	Nonlinear biochemical dynamic model with varying metabolite concentration		E
VARMOL	3	78	2	0	0	Gas phase tubular reactor	[234]	S5
VARVOL	4	154	3	0	0	Variable volume fermentation	[126]	S5
VDPOL0	2	16	2	0	0	Van der Pol equation, electrical circuit	[199]	S0
VDPOLM	2	18	2	0	0	Van der Pol equation, electrical circuit	[199]	S5
VERT_CHL	8	30	4	6	6	Injected fluid flow in a long, vertical channel (4th order BVP)	[11]	S5
VESSEL	2	5	1	0	0	Vessel adsorption		E
W_WATER1	4	15	2	0	0	Anaerobic wastewater treatment, monod model without decay	[346]	S5
W_WATER2	4	50	2	0	0	Anaerobic wastewater treatment, monod model with substrate inhibition	[346]	S5
WASTEWAT	3	14	5	0	0	Anaerobic wastewater treatment	[362]	S5
WAVE_D1	1	100	10	0	0	Hyperbolic wave equation with exact solution, PDE discretized (Dirichlet)	[420]	S5
WAVE_D2	1	100	10	0	0	Hyperbolic wave equation with exact solution, PDE discretized (Neumann)	[420]	X
WEEDS	8	16	2	0	0	Growth of two weeds in competition	[402]	E
WEIBEL	3	13	1	0	0	Ill-conditioned academic equation		S5
WINDSHEAR	10	7	6	35	0	Optimal take-off trajectories under wind shear	[331]	E
YEASTOSC	1	15	5	0	0	Oscillating continuous Baker's yeast culture	[126]	S5
ZOOPLANK	2	303	3	0	0	Growth of phytoplankton and zooplankton		S5

10.5 Differential Algebraic Equations

As before, we have r data sets (t_i, c_j, y_{ij}^k) with $l = l_t l_c r$, and l weights w_{ij}^k . Again, weights can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for state variables for which E data do not exist. The subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation.

The data fitting function $h(p, y(p, t, c), z(p, t, c), t, c)$ depends on a concentration parameter c and in addition on the solution $y(p, t, c)$ and $z(p, t, c)$ of a system of m_d differential and m_a algebraic equations

$$\begin{aligned} \dot{y}_1 &= F_1(p, y, z, t, c) \quad , \quad y_1(0) = y_1^0(p, c) \quad , \\ &\dots \\ \dot{y}_{m_d} &= F_{m_d}(p, y, z, t, c) \quad , \quad y_{m_d}(0) = y_{m_d}^0(p, c) \quad , \\ 0 &= G_1(p, y, z, t, c) \quad , \quad z_1(0) = z_1^0(p, c) \quad , \\ &\dots \\ 0 &= G_{m_a}(p, y, z, t, c) \quad , \quad z_{m_a}(0) = z_{m_a}^0(p, c) \quad . \end{aligned}$$

Without loss of generality, we assume that the initial time is zero. Now $y(x, t, c)$ and $z(x, t, c)$ are solution vectors of a joint system of $m_d + m_a$ differential and algebraic equations (DAE). The initial values of the differential equation system $y_1^0(p, c), \dots, y_{m_d}^0(p, c)$ and $z_1^0(p, c), \dots, z_{m_a}^0(p, c)$ may depend on one or more of the system parameters to be estimated, and on the concentration parameter c .

The system of differential equations is called an index-1-problem or an index-1-DAE, if the algebraic equations can be solved with respect to z , i.e., if the matrix

$$\nabla_z G(p, y, z, t, c)$$

possesses full rank. In this case, consistent initial values are can be computed internally.

The resulting parameter estimation problems is

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} \sum_{j=1}^{l_c} (w_{ij}^k (h_k(p, y(p, t_i, c_j), z(p, t_i, c_j), t_i, c_j) - y_{ij}^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0 \quad , \quad j = 1, \dots, m_e \quad , \\ & g_j(p) \geq 0 \quad , \quad j = m_e + 1, \dots, m_r \quad , \\ & p_l \leq p \leq p_u \quad . \end{aligned}$$

We assume that the model functions $h_k(p, y, z, t, c)$ and $g_j(p)$ are continuously differentiable functions of p , $k = 1, \dots, r$ and $j = 1, \dots, m_r$, and that the state variables $y(p, t_i, c_j)$ and $z(p, t_i, c_j)$ are smooth solutions subject to p . All test problems based on differential algebraic equations are listed in Table B.5, where constraint counts are omitted.

Table B.5. Differential Algebraic Equations

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_d</i>	<i>m_a</i>	<i>background</i>	<i>ref</i>	<i>data</i>
2LKC_ROB	5	80	12	5	Two-link planar robot with constraints	[11]	S5
AEROSOL	4	29	2	2	Substrate concentration in two-phase aerosol devices		E
AEROSOLS	5	36	2	2	Substrate concentration in two-phase aerosol devices, simulated data		S1
BATCH	9	128	6	3	Isothermal batch reactor, slow and fast reactions	[41]	S5
BATCH_E	9	204	6	4	Isothermal batch reactor, slow and fast reactions, two data sets	[41]	E
BATCH_E1	9	131	6	4	Isothermal batch reactor, slow and fast reactions, data for 40 deg C	[41], [497]	E
BATCH_E2	9	84	6	1	Isothermal batch reactor, slow and fast reactions, data for 67 deg C	[41], [497]	E
BATCH_E3	9	124	6	1	Isothermal batch reactor, slow and fast reactions, data for 100 deg C	[41], [497]	E
BATCH_F1	9	120	6	4	Isothermal batch reactor, slow and fast reactions, one data set (313,15)	[41]	E
BATCH_F2	9	84	6	4	Isothermal batch reactor, slow and fast reactions, one data set (340,15)	[41]	E
BATCH_F3	9	204	6	4	Isothermal batch reactor, slow and fast reactions, two data sets	[41]	E
BATCH_F4	9	328	6	4	Isothermal batch reactor, slow and fast reactions, three data sets	[41]	E
BATCH_X1	17	120	6	4	Isothermal batch reactor, slow and fast reactions, experimental design	[41]	E
BATCH_X2	13	78	6	4	Isothermal batch reactor, slow and fast reactions, experimental design	[41]	ED
BATCHREA	5	66	6	1	Batch reactor	[75]	S5
BOND	4	24	2	1	Transition of photon in a hydrogen-hydrogen bond	[273]	S5
BUBBLEC	3	72	8	5	Bubble point calculation for a batch distillation column	[234]	S5
CAT_SD	6	300	4	4	Catalyst semiconductor		E
CELLS	5	120	3	2	Cultivation of isolated plant cells in suspension culture	[345]	S1
CONDENS	2	114	1	5	Condensation of methanol with constant volume	[372]	S5
COPPER1	5	277	6	1	Multilayer model for moisture adsorbed on copper		E
DAE_EX	2	12	3	0	DAE with singularity		S5
DAE_I1	2	100	4	1	Academic example, index-1-formulation		S0
DAE_I2	2	40	4	1	Academic example, index-2-formulation		S0
DAE_I3	2	40	4	1	Academic example, index-3-formulation		S0

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_d</i>	<i>m_a</i>	<i>background</i>	<i>ref</i>	<i>data</i>
DAE_IN2	2	26	2	1	System of three differential algebraic equations with index 2	[11]	S5
DAE_IN2X	4	30	2	1	System of 3 differential algebraic equations with index 2, exact solution	[11]	X
DAE_SYS	2	20	1	2	Particle diffusion and reaction (2nd order BVP)	[11]	S1
DISTILL	3	66	22	22	Distillation column for two substances	[193]	S1
DISTILL3	9	205	106	53	Distillation column for three substances		S1
EVAPOR	3	38	3	10	Evaporation of benzol with constant volume	[316]	S5
EXOBATCH	4	80	5	6	Batch reactor with strongly exothermic reactions and cooling jacket	[525]	S5
HYDRODYN	2	15	2	1	Unipolar hydrodynamic model for semiconductors in the isotropic case	[11]	S5
LKIN_BR	2	34	2	0	Simple linear compartment model with two break points		S5
MARBLE	2	202	6	1	Movement of marble in a bowl		S5
MEM_WIRE	5	40	3	1	Optimal form of shape memory wires		S1
P_IDENT	6	138	2	1	Identification of parameters, academic example	[315]	S5
P_IDENT1	4	138	2	1	Identification of parameters, academic example, reduced parameter set	[315]	S5
P_IDENT2	4	19	2	1	Identification of parameters, academic example, one data set and u1=u2=1	[315]	S5
P_IDENT3	8	19	2	1	Identification of parameters, academic example, one data set and experimental design	[315]	ED
PENDU_CS	2	80	8	2	Plain pendulum, index-3-formulation with two algebraic equations		S1
PENDU_I1	2	80	4	1	Plain pendulum, index-1-formulation		S1
PENDU_I2	2	80	4	1	Plain pendulum, index-2-formulation		S1
PENDU_I3	2	80	4	1	Plain pendulum, index-3-formulation		S1
PENDU_IV	2	80	4	3	Plain pendulum, consistent initial values computed internally		S1
PENDU_PD	2	80	4	3	Plain pendulum, projected descriptor formulation		S1
PENDULUM	2	80	4	1	Plain pendulum		S1
PHOSPH_A	3	54	3	2	Chemical reaction, phosphorescence		S5
RESPIR	3	40	1	1	Human respiratory system	[529]	S5
SCARA	4	204	8	2	Scara robot with four parameters and given control function	[219]	N5
SCARA_X	24	66	8	2	Scara robot with experimental design	[219]	ED
SHOCK	4	60	6	3	Reaction zone in detonating explosives	[130]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>m_d</i>	<i>m_a</i>	<i>background</i>	<i>ref</i>	<i>data</i>
TRAN_A_B	4	102	2	1	Transport of substrate by Michaelis-Menten kinetics		S1
TRANSIST	3	11	3	2	Transistor amplifier, highly oscillating		E
TRUCK	3	84	22	1	Truck model (multibody system)	[479]	S1
TUBULAR	8	42	2	2	Stationary tubular reactor with cooling wall	[327]	S5
URETHAN	15	30	3	3	Urethan reaction in a semi batch reactor with two feed vessels	[29]	E
URETHANW	55	164	3	3	Urethan reaction in a semi batch reactor, experimental design with weights	[29]	ED
URETHANX	55	40	3	3	Urethan reaction in a semi batch reactor, experimental design	[29]	ED
VAS_ADSP	4	14	2	1	Vascular adsorption		S5
VAS_ADSS	13	34	6	3	Vascular adsorption with three experimental data sets		E
VDPOL	2	10	1	1	Van der Pol equation, electrical circuit		S0

10.6 Partial Differential Equations

Now we proceed from r data sets

$$(t_i, y_i^k), \quad i = 1, \dots, l_t, \quad k = 1, \dots, r, \quad ,$$

where l_t time values and $l = l_t r$ corresponding measurement values are defined. Moreover, we assume that l weights w_i^k are given, which can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for state variables for which E data do not exist. The subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation. The additional independent model variable c called concentration in the previous models, is not taken into account for simplicity.

The system of partial differential equations under consideration is

$$\begin{aligned} \dot{u}_1 &= F_1(p, u, u_x, u_{xx}, v, x, t) \quad , \\ &\dots \\ \dot{u}_{n_p} &= F_{n_p}(p, u, u_x, u_{xx}, v, x, t) \end{aligned}$$

with state variable $u = (u_1, \dots, u_{n_p})^T$. We denote the solution of the system of partial differential equations by $u(p, x, t)$ and $v(p, t)$, since it depends on the time value t , the space value x , and the actual parameter value p . v denotes the additional coupled differential variable. To simplify the notation, flux functions are omitted.

Initial and boundary conditions may depend on the parameter vector to be estimated. Since the starting time is assumed to be zero initial values, have the form

$$u(p, x, 0) = u_0(p, x)$$

and are defined for all $x \in (x_L, x_R)$. For both end points x_L and x_R we allow Dirichlet or Neumann boundary conditions

$$\begin{aligned} u(p, x_L, t) &= u^L(p, v, t) \quad , \\ u(p, x_R, t) &= u^R(p, v, t) \quad , \\ u_x(p, x_L, t) &= \hat{u}^L(p, v, t) \quad , \\ u_x(p, x_R, t) &= \hat{u}^R(p, v, t) \end{aligned}$$

for $0 < t \leq T$, where T is the final integration time, for example the last E time value t_{l_t} . We do not require the evaluation of all boundary functions. Instead, a user may omit some of them depending upon the structure of the PDE model, for example whether second partial derivatives exist in the right-hand side or not.

In addition, the partial differential equation may depend on the solution of a system of ordinary differential equations $v \in \mathbb{R}^{n_c}$ given in the form

$$\dot{v}_j = G_j(p, u(p, x_j, t), u_x(p, x_j, t), u_{xx}(p, x_j, t), v, t)$$

for $j = 1, \dots, n_c$, where $u(p, x, t)$ is the solution vector of the partial differential equation. Here x_j are any x -coordinate values where the corresponding ordinary differential equation is coupled to the partial one. Some of these values may coincide. When discretizing the system by the method of lines, they are rounded to the nearest neighboring grid point. The corresponding initial values

$$v(p, 0) = v_0(p)$$

may depend on the parameters to be estimated.

Each set of E data is assigned a spatial variable value $x_k \in (x_L, x_R)$, $k = 1, \dots, r$, where r denotes the total number of measurement sets. Some or all of the x_k -values may coincide, if different measurement sets are available at the same local position. Since partial differential equations are discretized by the method of lines, the fitting points x_k are rounded to the nearest spatial grid point or line, respectively.

The resulting parameter estimation problems is

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{t_i} (w_i^k (h_k(p, u(p, x_k, t_i), u_x(p, x_k, t_i), \\ & \quad \quad \quad u_{xx}(p, x_k, t_i), v(p, t_i), t_i) - y_i^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0, \quad j = 1, \dots, m_e, \\ & g_j(p) \geq 0, \quad j = m_e + 1, \dots, m_r, \\ & p_l \leq p \leq p_u. \end{aligned}$$

It must be assumed that all model functions $h_k(p, u, u_x, u_{xx}, v, t)$ and $g_j(p)$ are continuously differentiable subject to p for $k = 1, \dots, r$ and $j = 1, \dots, m_r$, also the state variables and their spatial derivatives $u(p, x, t)$, $u_x(p, x, t)$, $u_{xx}(p, x, t)$, and $v(p, t)$.

All test problems of our collection based on time-dependent, one-dimensional partial differential equations are listed in Table B.6. Not listed are the number of integration areas, switching times, and structure of the boundary conditions. Equality constraints do not exist in this case, and m_e is therefore omitted.

Table B.6. Partial Differential Equations

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
2AREAS	5	24	1	0	0	Diffusion (Fick's law) in two areas	[421]	S5
2AREAS_C	4	101	1	1	0	Diffusion (Fick's law) in two areas and coupled ODE		S5
2MEMBRAN	4	26	1	3	0	Two membranes with explicitly modeled transition		E
ACCRET	1	40	2	0	0	Thermal equilibrium curves in Keplerian accretion disks	[381]	S5
ADV_DIFF	2	150	3	0	0	Advection-diffusion equation with Riemann initial data	[248]	S1
ADV_DOM	2	150	1	0	0	Advection dominated equation, wave from left boundary	[491]	S5
ADV_DOMS	2	48	3	0	0	Advection dominated equation, wave from left boundary, with sensitivity equations	[491]	S1
ADV_VC	2	60	1	0	0	Advection equation with variable coefficient	[498]	X
ADV_VTC	3	90	2	0	0	Advection equations with variable time coefficient	[498]	S5
ADVEC_2N	3	50	1	0	0	Nonlinear unsteady advection (n=2)	[533]	S1
ADVEC_5N	3	20	1	0	0	Nonlinear unsteady advection (n=5)	[533]	S5
ADVEC_CP	2	20	2	2	0	Two coupled advection equations with periodic boundary conditions	[266], [533]	X
ADVEC_LU	4	35	1	0	0	Linear unsteady advection-diffusion	[533]	S5
ADVEC_PB	1	50	1	1	0	Advection with periodic boundary condition	[266], [533]	S5
ADVEC_S	4	5	1	0	0	Linear steady advection-diffusion with source term	[533]	S5
ADVECT	1	171	1	0	0	Advection equation, first-order hyperbolic PDE	[420]	X
ADVECT_N	1	128	1	0	0	Advection equation with a nonlinear source term	[376]	S5
ADVECT_R	1	180	1	0	0	Advection equation with right boundary value	[420]	X
ADVECT_S	1	128	2	0	0	Advection with a nonlinear source term and sensitivity equations	[376]	S5
ADVECT2	2	190	2	0	0	Two advection equations (different flux directions)	[420]	X
ADVECT2A	2	190	2	0	0	Two advection equations (different flux directions, two areas)	[420]	X
AFFIN	4	60	2	0	0	Affinity membrane separation of a protein solution	[76]	S5
AIR_FLOW	1	32	3	0	0	Flow of air in shock-tube (Euler equations of gas dynamics, Riemann data)	[376]	S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
AL_ALLOY	2	20	1	0	0	Finite heat-conducting in aluminium alloy	[32], [142]	E
ALDRIN_P	4	10	1	0	0	Diffusion with chemical reaction		E
ARA_YARN	3	120	1	0	0	Water penetration into an aramide yarn	[497]	S5
AX_DIFF	2	11	1	0	0	Continuous tubular reactor with axial diffusion	[85]	S0
BALLONS5	5	26	1	5	0	Diffusion of air through skins of 5 bowls		E
BEEETLES	2	80	1	0	0	Flea beetles in cultivated linear arrays of collard patches (insect dispersal study)	[24]	S0
BINDSITE	6	50	6	0	0	Plasma, extravascular, and binding sites with two injections		E
BIOFILM	3	20	2	2	0	Double substrate biofilm reaction	[126]	S5
BLD_BRN	10	14	1	6	0	Blood-brain barrier		E
BLOW_UP	1	49	1	0	0	Degenerated parabolic equation with blow-up	[155]	S5
BRAIN	3	90	1	0	0	Transport phenomena in brain tissue	[24]	S5
BRINE	1	15	2	0	0	Brine transport in porous media	[560]	X
BRUSS2D	2	220	42	0	0	Two-dimensional brusselator	[199]	S5
BRUSSEL	3	120	2	0	0	Brusselator with diffusion	[151], [433]	S5
BSE	4	10	1	0	0	Black-Scholes equation governing price of derivative security	[48], [554]	S1
BUBB_BIO	1	10	3	0	0	Bubble column bio-reactor	[344]	S5
BUBBLE	2	108	2	0	0	Dynamic oxygen uptake of water in bubble column		S5
BURGER	1	24	1	0	0	Parabolic Burger's equation with exact solution	[480]	X
BURGER_E	1	120	1	0	0	Viscous Burger's equation with exact solution, mue=0.01	[420], [433]	X
BURGER_F	1	20	1	0	0	Viscous Burger's equation with exact solution, mue=1	[420], [433]	X
BURGER_I	3	32	1	0	0	Burger's equation in the inviscid limit	[556], [376]	S5
BURGER_R	1	44	1	0	0	Burger's equation with Riemann initial data	[480]	X
BURGER_X	3	50	1	0	0	Viscous Burger's equation with exact solution, eps=0.0005		X
BURST	1	105	2	0	0	Crisis induced intermittent bursting in reaction-diffusion chemical systems	[139], [135]	S5
CALIBR	12	195	4	4	0	Substrate diffusion through two areas		E
CARRIER	3	12	3	2	0	Diffusion through membrane based on carrier effect		S5
CD_TRANS	1	28	1	0	0	Convective-dispersive transport equation with nonlinear reactions	[256]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
CN_PLAS	2	30	5	0	0	Magnetohydrodynamic model of a charge-neutral plasma	[521]	S5
CNT_CUR1	4	80	2	0	0	Counter-current separation of fluid phase concentrations with phase equilibrium	[379]	S5
COLLAGEN	2	8	2	0	0	Drug release from collagen matrices	RBKLF01	S5
COMP_MED	3	10	1	0	0	Infinite composite medium	[97]	S1
CON_DIV1	1	90	1	0	0	Periodic convection dominated diffusion	[319]	X
CON_DIV2	1	63	1	0	0	Periodic convection dominated diffusion	[319]	X
CONTAMIN	3	24	4	4	0	Contamination of aqueous solutions	[433]	S5
COS_PROF	1	171	1	0	0	Convection equation, propagation of a cosinus profile	[420]	S5
CPA_PLAS	3	12	1	2	0	CPA plaster with three diffusion areas		E
CPL_ADV	2	90	2	2	0	Two coupled linear advection equations	[291]	S1
CRYSTAL	4	51	2	0	0	Crystal dissolution fronts in flows through porous media	[254]	S5
CSE	2	41	2	0	0	Cubic Schroedinger equation with one soliton	[421]	S5
CTFLOW_P	1	44	2	0	0	Two incompressible counter-current flows of binary liquid mixture with permeable wall	[339]	S5
CTRL_WAV	20	40	2	0	0	Optimal control problem, wave equation		S5
CUBIC	1	10	1	0	0	Cubic conservation law with Riemann data	[212]	S5
DAMBREAK	2	30	2	0	0	Idealized dam break, sudden and complete removal	[491]	S5
DBVP	2	9	1	0	0	Dirichlet boundary value problem with dominating heat source	[91]	S5
DC_TUBE	1	7	1	0	0	Diffusion-convection in a tube		S1
DEHYDRO	4	60	2	0	0	Dehydrogenization of ethylbenzene to styrene in a tubular reactor	[534]	S05
DERMAL	10	25	2	4	0	Transdermal skin model in two areas with transitions	[492]	E
DESIGN	1	56	1	0	0	First-order hyperbolic PDE, inhomogeneous part	[513]	S1
DIALYSI1	4	179	1	1	0	Dialysis membrane with exponential diffusion coefficient, long term experiment		E
DIALYSI2	2	96	1	2	0	Substrate diffusion through dialysis membrane		E
DIALYSI3	3	100	1	1	0	Substrate diffusion through dialysis membrane with two areas		E
DIALYSI4	8	298	3	3	0	Substrate diffusion through dialysis membrane with 2 areas, 3 data sets		E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
DIALYSI5	7	298	3	3	0	Substrate diffusion through dialysis membrane with 2 areas, 3 data sets		E
DIFF_ID	4	151	1	0	0	Diffusion problem with Dirichlet and Neumann boundary conditions		E
DIFF_ADS	3	50	2	0	0	Diffusion and absorption reaction		E
DIFF_CON	2	40	1	0	0	Diffusion-convection problem with discontinuous coefficients	[480]	S5
DIFF_ETH	2	50	1	0	0	Diffusion of ethanol in water	[534], [220]	S5
DIFF_NLB	4	35	1	0	0	Nonlinear diffusion with nonlinear boundary condition	[480]	S5
DIFF_P	3	12	1	2	0	Flow with diffusion through tube wall		S5
DIFFPT	3	10	1	0	0	Diffusion and partitioning in biological systems, non-continuous transition	[335]	S1
DIFFPT1	3	20	1	0	0	Diffusion and partitioning in biological systems, exponential initial values	[335]	S5
DIFFPT2	3	10	1	0	0	Diffusion and partitioning in biological systems, non-continuous transition	[335]	S1
DIFFPT3	4	20	1	0	0	Diffusion and partitioning in biological systems, continuous flux transition	[335]	S5
DIFFPT4	4	20	1	0	0	Diffusion and partitioning in biological systems, continuous flux	[335]	S5
DIFFREA	3	292	1	1	0	Diffusion and reaction in solid phase, coupled ODE		E
DIFFREA1	9	292	1	0	0	Diffusion and reaction in solid phase		E
DIFFUS	1	81	1	0	0	Diffusion equation with constant parameters		S5
DIGESTOR	2	31	1	0	0	Fixed-bed anaerobic digester		S5
DISADVFL	2	22	1	0	0	Dispersive advective flow		E
DISRE	3	30	1	0	0	Non-isothermal tubular reactor with axial dispersion	[234], [433]	S5
DISRET	2	12	2	0	0	Non-isothermal tubular reactor with axial dispersion and non-constant reaction term	[234], [433]	S5
DRY	2	46	1	2	0	Drying of a solid	[234]	S1
ECOLOGY	2	60	2	0	0	Population ecology with planktonit predator-prey and crowding	[270]	S5
ELASTIC	1	20	2	0	0	Elastic model in conservative form with discontinuity	[96]	S5
ELEC_DYN	3	20	2	0	0	Electrodynamical application	[50]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
ELECTRO	3	38	2	0	0	Electrodynamic model	[433]	S5
ELLIPTIC	3	42	1	0	0	Elliptic test problem	[480]	S5
ENERGY	2	5	1	0	0	Tubular reactor based on energy equation	[177]	S5
ENZDYN	3	22	2	0	0	Dynamic diffusion and enzymatic reaction	[234]	S5
EXR14_1	3	10	1	0	0	Signaling problem	[292]	S5
EXR14_2	1	13	1	0	0	Initial value problem with non-continuous initial value	[292]	S5
EXR14_3	3	14	1	0	0	Initial-boundary value problem	[292]	S5
EXR15_4	1	30	2	0	0	Traveling wave	[292]	S5
EXR21_1	2	100	1	0	0	Initial value problem	[292]	S5
EXR22_1	2	10	1	0	0	Initial value problem	[292]	S5
EXR23_1	2	10	1	0	0	Initial value problem	[292]	S5
EXR24_1	2	13	1	0	0	Nonlinear initial value problem	[292]	S5
EXR25_1	2	20	1	0	0	Initial value problem	[292]	S5
EXR25_2	2	10	1	0	0	Nonlinear initial value problem	[292]	S5
EXR25_6	2	8	1	0	0	Nonlinear initial value problem	[292]	S5
EXR31_4	2	11	1	0	0	Initial value problem with shock formation	[292]	X
EXR32_1	2	11	1	0	0	Initial value problem, breaking wave	[292]	S5
EXR32_4	2	10	1	0	0	Initial-boundary value problem	[292]	S5
EXR32_7	2	28	1	0	0	Initial value problem, formation of wave	[292]	S5
EXR32_8	2	8	1	0	0	Initial value problem with shock path	[292]	S5
EXR32_9	2	24	1	0	0	Initial value problem with two shocks merging into one shock	[292]	S5
EXR34_12	2	10	1	0	0	Riemann problem	[292]	S5
EXR42_2	2	10	1	0	0	Initial-boundary value diffusion problem	[292]	S5
EXR43_2	2	10	1	0	0	Nonlinear diffusion problem	[292]	S5
EXR43_3	2	10	1	0	0	Diffusion problem	[292]	S5
EXR52_5	2	28	2	0	0	Hyperbolic initial value problem, linear	[292]	S5
EXR64_4	2	10	1	0	0	Diffusion problem	[292]	S5
EXR65_5	1	10	1	0	0	Diffusion problem (no global solution for $a_i 1$)	[292]	S5
FILTWASH	2	20	1	0	0	Filter washing	[234]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
FINAG	2	246	2	0	0	Nerve conduction	[348]	S5
FIX_BED	2	16	2	0	0	Catalytic fixed bed reactor	[392]	S5
FIXBED	3	48	2	0	0	Catalytic fixed bed reactor with one exothermal reaction	[516], [133]	S1
FLAME	2	48	2	0	0	Dwyer-Sanders flame propagation model	[127], [527], [433]	S1
FLOW	2	18	1	0	0	Isothermal laminar-flow tubular reactor	[514]	S5
FLOW_PMD	1	80	1	0	0	Flow through porous media with degenerate initial values	[517]	S1
FLOW_PMW	3	120	1	0	0	Flow through porous media with waiting time	[517]	S5
FLUID	3	20	2	0	0	Diffusion (Fick's law)	[421]	S5
FOX	2	189	3	0	0	Rabies distribution of fox population	[347]	S1
FRONT	2	60	2	0	0	Flame propagation model with non-constant moving front	[377]	S5
G_HILL	2	33	1	0	0	Convection-diffusion of a Gaussian hill	[121]	S5
GAS_BUBB	2	180	1	14	0	Non-viscous gas bubble in oil with diffusion		S5
GAS_CONV	3	243	1	0	0	Gas convection		E
GAS_DIF1	1	8	1	2	0	One-dimensional gas diffusion in a column		E
GAS_DIF2	1	15	1	2	0	One-dimensional gas diffusion in a column		E
GLACIER	2	45	1	0	0	Glacier growth with conservation of mass and momentum, incompressible flow	[248]	S5
GLYCO	2	62	2	0	0	Glycolysis reaction-diffusion model with autocatalytical growth of species	[12]	S1
GROUND_W	5	20	1	0	0	Saturation of ground water (Richards equation)	[518], [433]	E
GROWTH	2	21	1	0	0	Logistic model of population growth (Fisher's equation)	[511]	S5
HEAT	1	6	1	0	0	Heat equation	[420]	S5
HEAT_B	6	36	1	0	0	Heat equation, break points and two integration areas with transition condition		S5
HEAT_BD3	2	32	1	0	0	Nonlinear heat equation, boundary conditions of third type		S5
HEAT_CD	2	10	1	0	0	One-dimensional heat conduction	[423]	S5
HEAT_CF	5	11	1	0	0	Heat transfer in a circular fin	[423], [43]	S5
HEAT_CON	3	76	1	0	0	Heat transfer in cylinder with heat loss by convection	[534]	S5
HEAT_CW	2	10	1	0	0	Graetz problem with constant wall heat flux	[423]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
HEAT_CYL	2	50	1	0	0	Cylindrical heat transfer	[480]	S5
HEAT_EX	3	20	1	0	0	Tubular heat exchanger	[423]	S5
HEAT_I	4	40	1	0	0	Heat equation, two integration areas with transition condition		S5
HEAT_MS	2	9	2	0	0	Heat transport equation at the microscale (3rd order)	[545]	X
HEAT_NLB	2	10	1	0	0	Heat equation with nonlinear boundary condition of Stefan-Boltzmann type	[510]	E
HEAT_S1	4	21	1	0	0	Heat equation with redundant parameters	[420], [433]	S5
HEAT_S2	3	11	1	0	0	Heat equation with redundant parameters	[420], [433]	S5
HEAT_SEN	3	40	4	0	0	Heat conduction with full sensitivity equations		S5
HEAT_SX	1	20	2	0	0	Heat equation with one sensitivity equation and exact solution		X
HEAT_TDC	3	42	1	0	0	Heat diffusion with time-dependent diffusion parameter		S5
HEAT_X	1	99	1	0	0	Heat equation with exact data and maximum norm		X
HOT_SPOT	2	110	1	0	0	'Hot Spot' problem from combustion theory	[527], [433]	S5
HUMID	2	33	3	0	0	Humidification column of porous medium	[420]	S1
HYDRO	1	32	2	0	0	St. Venant equation for fluid dynamics of hydro systems	[190]	S5
HYDRO_2C	6	20	2	0	0	St. Venant equation for fluid dynamics of hydro systems, two serial channels	[190]	X
HYDRO_3S	6	303	6	0	0	St. Venant equation for fluid dynamics of hydro systems, 3-star		none
HYDRO_FX	1	32	2	0	0	St. Venant equation for fluid dynamics of hydro systems, flux formulation	[190]	S5
HYG_POLY	3	74	1	0	0	Diffusion of water into a hygroscopic polymer		E
HYGROS	3	9	1	0	0	Diffusion of water through boundary layer of hygroscopic material and air		E
HYP_PBC	2	20	1	1	0	Hyperbolic equation with periodic boundary conditions	[12]	S5
HYP2ND	1	15	2	0	0	Hyperbolic equation of second order, alternating cosine waves		S5
HYPER	2	198	2	0	0	System of two advection equations, first-order hyperbolic PDEs		S5
HYPERBO1	2	90	2	0	0	Hyperbolic test system	[23]	S5
HYPERBO2	2	90	2	0	0	Hyperbolic test system	[23]	S5
HYPERBO3	2	90	2	0	0	Hyperbolic test system	[23]	S5
HYPERBO4	2	90	2	0	0	Hyperbolic test system	[23]	S5

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
HYPERBO5	3	90	2	0	0	Hyperbolic test system	[23]	S5
IN_LAYER	4	42	2	0	0	Catalyst with inert layers (diffusion, absorption, desorption)		S5
INTEG	3	25	1	0	0	Population dynamics with integro-differential equation	[376], [433]	E
INTERF1	2	10	1	0	0	System with interface (not modeled)	[433]	S5
INTERF2	2	18	1	0	0	System with interface	[433]	S5
INV_PROB	10	20	1	0	9	Inverse problem in heat conduction	[205]	S5
IONTRAN1	2	101	1	2	0	Ion transport through membrane by diffusion		S1
IONTRAN3	3	96	1	1	0	Ion transport through membrane with Langmuir isotherm for sorption		E
ISOTHRM1	10	34	2	0	9	Reactive solute transport, advective-dispersive transport	[232], [231]	S5
ISOTHRM2	4	20	1	0	3	Reactive solute transport, advective-dispersive transport (Freundlich version)	[232], [231]	S5
JONTO	4	17	1	1	2	Optimal control of iontophoresis with three membranes		none
KILN	10	22	1	0	0	Heating a probe in a kiln	[110]	E
KIN_SORP	2	101	2	0	0	Kinetic sorption by advection-dispersion		S5
LAM_FLOW	1	10	1	0	0	Unsteady laminar flow in a circular tube	[423], [229]	S5
LAPLACE	1	20	20	0	0	Laplace equation (elliptic)	[420]	X
LDCP	1	100	1	0	0	Linear diffusion-convection equation	[391], [379]	S5
LIN_ADV	4	90	1	0	0	Linear advection problem, highly nonlinear initial condition	[238]	S5
LIN_HC	3	140	1	0	0	Linear heat conduction	[3]	X
LIN_HYP1	2	25	1	0	0	First-order linear hyperbolic equation	[540]	X
LIN_HYP2	3	70	1	0	0	First-order linear hyperbolic equation with interface	[540]	X
LIN_HYP3	2	212	1	0	0	First-order linear hyperbolic equation with variable velocity field	[540]	X
LNCHROM1	3	1	1	0	0	Nonlinear chromatographic system	[231]	S5
LNCHROM2	2	282	4	0	0	Nonlinear chromatographic system		S5
LOSSLESS	2	20	2	0	0	Lossless electric transmission line	[421]	X
LUNG	5	33	1	4	0	Protein application in lung with decomposition		E
MALTDX10	14	114	10	10	0	Drying of maltodextrin in a convection oven, simultaneous fitting of 10 data sets	[154]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
MALTODEX	5	12	1	1	0	Drying of maltodextrin in a convection oven, first data set	[154]	E
MASS_TRA	1	25	1	0	0	Mass transfer with simultaneous convection and diffusion	[423]	S5
MEDAKZO	2	21	2	0	0	Medical Akzo-Nobel problem	[289]	S5
MEMSEP	3	25	2	0	0	Affinity membrane separation of a protein solution	[75]	S5
MEMBRANE	3	20	2	0	0	Diffusion through a membrane	[351]	S5
MILL1	1	80	1	0	0	Rolling mill cooling, constant psi in boundary condition	[433]	S0
MILL2	5	40	1	0	0	Rolling mill cooling, variable phi in boundary condition	[433]	S1
MILL3	2	80	1	0	0	Rolling mill cooling, estimating heat transfer coefficients	[433]	S1
MOLDIFF	2	10	1	0	0	Molecular diffusion (boundary value problem)	[330]	E
MOVFRONT	3	126	1	0	0	Moving front (Burger's equation)	[3]	S5
MX_ENTRO	9	80	1	0	0	Maximum entropy method, advection-diffusion equation		S5
MZ_FURN	5	63	2	0	0	Multizone electrical furnace for production of integrated circuits	[555]	S05
N_CONVEX	1	36	1	0	0	Hyperbolic test problem of Shu and Osher, nonconvex flux	[237]	S5
NDYN	3	20	2	0	0	Nitrogen and ammonium dynamics in forest soils	[72], [433], [150]	E
NERVE	4	200	2	0	0	Nerve pulse	[528]	S5
NL_HEAT	3	18	1	0	0	Nonlinear heat equation	[511]	S5
NL_PDE	3	10	1	0	0	Highly nonlinear PDE with exact solution	[421]	X
NL_PDE1	1	10	1	0	0	First order nonlinear PDE with exact solution		S5
NL_TRANS	2	11	1	1	0	Nonlinear transport equation developing a shock (Burger), periodic boundary	[428]	S5
NL2_SORP	3	38	2	0	0	Nonlinear two-point sorption		E
NLIN_2ND	2	100	1	0	0	Nonlinear second-order partial derivatives		S5
NLINPDE	2	100	2	0	0	Two nonlinear PDE's with exact solution	[420], [313]	X
NLSE	4	18	2	0	0	Nonlinear Schroedinger equation, exact soliton solution (complex)		S1
NOISE	2	20	1	0	0	Nonlinear deblurring and noise removal	[322]	S5
NON_AD	1	60	1	0	0	Nonlinear advection-diffusion equation	[248]	S5
OBSTACLE	2	20	2	0	0	Shallow water flow over an obstacle	[290]	S1
ONESTEP	2	130	2	0	0	One-step reaction with diffusion and non-unit Lewis number	[3]	S1
OSC_SOL	2	20	3	0	0	Oscillatory solution of hyperbolic PDE	[140]	S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
PACK_BED	2	64	4	0	0	Fluid through a packed bed with adsorption/desorption of two components		S5
PAR_CTRL	2	10	1	0	0	Parabolic optimal control problem	[336]	X
PAR_SIN	2	84	1	0	0	Parabolic PDE with inhomogeneous sinus-term	[397], [380]	X
PARAB1	3	8	1	0	0	Parabolic equation, identifiability test	[24]	S0
PARAB2	3	60	2	0	0	Parabolic equation, identifiability test	[24]	S0
PARAB3	2	30	1	0	0	Parabolic equation, identifiability test	[24]	S0
PARAB4	3	30	1	0	0	Parabolic equation, identifiability test	[24]	S0
PARAB5	3	10	1	0	0	Parabolic equation, identifiability test	[24]	S0
PARAB6	6	27	1	0	0	Parabolic equation, identifiability test	[24]	X
PB_CTRL1	51	98	1	0	0	Parabolic optimal control problem	[172]	none
PB_CTRL2	11	48	1	0	0	Parabolic optimal control problem	[172]	none
PERIOCHP	2	12	3	0	0	Diffusion-mediated release from bulk degrading matrices in dental prostheses	[512]	E
PHYP_PBC	2	20	1	1	0	Parabolic-hyperbolic equation with periodic boundary conditions	[12]	S5
POLLUTN	8	28	4	0	0	SST pollution in the stratosphere	[480], [433]	S1
POLY_DYN	3	15	1	1	0	Chain length of polymerization process		S5
POLYMERI	9	45	12	0	0	Radical copolymerization of methylmethacrylat and styren	[460]	S1
POOL	3	28	2	1	0	Evaporation of vapor from a pool of liquid	[37]	S5
PORE	7	51	2	1	0	Diffusion through polymer pores		S5
QUENCH1	2	63	1	0	0	Degenerate nonlinear quenching	[472]	S5
QUENCH2	1	6	1	0	0	Degenerate nonlinear quenching	[472]	S5
REA_DIF1	1	45	1	0	0	Reaction-diffusion equation	[134]	S5
REA_DIF2	2	135	1	0	0	Reaction-diffusion equation	[134]	S5
RESERVOI	1	10	1	0	0	Reservoir simulation by the Buckley-Leverett equation	[521]	S5
RICH_EQU	4	120	1	0	0	Saturation of ground water (Richards equation)	[518], [400]	S1
RICH_LPT	5	202	1	1	0	Non-stationary fluid transport through porous media by Richards equation	[46]	S5
RICH_XEN	3	36	1	0	0	Saturation of ground water (Richards equation)	[518], [400]	S1

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
RIE_BND	4	20	3	0	0	Flow of air in shock-tube (Euler equations of gas dynamics), flux formulation	[376]	S1
RIE_CD	1	33	3	0	0	Riemann problem for Euler equations of a polytropic gas with contact discontinuity	[291]	S1
RIE_LAX	1	6	3	0	0	Riemann problem for Euler equations, formulation of Lax	[291]	S5
RIE_LD	1	33	3	0	0	Low density and internal energy Riemann problem for Euler equations	[291]	S1
RIE_SEWI	2	60	3	0	0	Riemann problem with shock entropy wave interaction for Euler equations	[238]	S5
RIE_SO	2	20	3	0	0	Riemann problem for Euler equations with Shu-Osher sine wave hitting shock	[291]	S5
RIE_SOD	1	6	3	0	0	Sod's Riemann problem for Euler equations of a polytropic gas	[291]	X
ROD	4	10	1	0	0	Rod of solid explosive		S1
SALINE	4	20	2	2	0	Diffusion of drug in a saline solution through membrane	[489]	S5
SE_PULSE	2	23	1	0	0	Advection of semi ellipse pulse	[163]	S5
SETTLER	3	10	1	0	0	Solid dynamics within settling zone	[277]	S5
SH_FRONT	2	20	1	0	0	PDE with sharp front, exact solution known	[114]	X
SHEAR	4	33	3	0	0	Shear band formation	[361], [151], [433]	S1
SIN_GOR1	2	80	2	0	0	Sine-Gordon equation, exact kink-soliton solution	[422]	X
SIN_GOR2	1	80	2	0	0	Sine-Gordon equation, exact kink-kink-collision solution	[422]	X
SINGSTEP	2	130	1	0	0	Single-step reaction with diffusion	[3]	S1
SKIN_X	5	25	2	4	0	Transdermal diffusion	[433], [54]	ED
SKIN1	10	25	2	4	0	Transdermal diffusion	[433], [54]	E
SKIN10	10	25	2	4	0	Transdermal diffusion (STEP 0: First trial with all parameters)	[433], [54]	E
SKIN11	4	25	2	4	0	Transdermal diffusion (STEP 1: Data fitting with significant parameters)	[433], [54]	E
SKIN12	6	25	2	4	0	Transdermal diffusion (STEP 3: Experimental design)	[433], [54]	E
SKIN13	6	25	2	4	0	Transdermal diffusion (STEP 4: Evaluate confidence intervals at initial design)	[433], [54]	E

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
SKIN14	6	25	2	4	0	Transdermal diffusion (STEP 5: Evaluate confidence intervals at final design)	[433], [54]	E
SKIN15	6	25	2	4	0	Transdermal diffusion (STEP 5: Experimental design with weights)	[433], [54]	E
SKIN16	3	3	2	4	0	Transdermal diffusion (STEP 7: Identifiability at three optimal weights)	[433], [54]	E
SKIN2	8	25	3	6	0	Skin model with association kinetics	[54]	E
SKIN3	3	25	2	4	0	Skin model, in vitro experiment, with perfect sink	[54]	E
SKIN3_X	6	68	2	4	0	Skin model, in vitro experiment, with perfect sink	[54]	ED
SKIN4	3	56	2	4	0	Transdermal diffusion	[54]	S1
SKIN5	7	25	2	4	0	Transdermal diffusion	[433]	E
SLAB	3	36	3	0	0	Dwyer-Sanders flame propagation model	[369]	S5
SLAB_CTR	20	10	1	0	16	Temperature control of a slab	[17]	none
SOIL	3	80	2	0	0	Diffusion of water through soil, convection and dispersion	[519], [8], [433]	E
SOLID	2	14	1	0	0	Heating of solid sphere	[16]	S5
SOLITON	2	20	2	0	0	Kink soliton (Sine-Gordon equation)		S5
SORP_IS1	2	11	1	0	3	Reactive solute transport, convective-diffusive transport (Freundlich)	[231]	S5
SORP_IS2	2	11	1	0	0	Reactive solute transport, convective-diffusive transport (Langmuir)	[231]	S5
SORP_IS3	4	22	1	0	3	Reactive solute transport, convective-diffusive transport	[231]	S5
SORPTION	3	41	1	1	3	Transport equation (diffusion and sorption), ground water flow with contamination		E
SOUND	2	10	1	0	0	Sound in tissue	[433], [54]	S1
SPHERE	2	16	1	0	0	Heat conduction in sphere with exothermic chemical reaction	[420]	S5
STAR_NET	3	30	3	1	0	Parabolic star net		S1
STARTBED	2	81	1	0	0	Diffusion		E
STARTUP	3	30	11	0	0	Startup phase of an automobile catalytic converter	[131]	S1
STEPHAN	2	110	1	1	0	One-phase Stephan problem	[37]	X

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_p</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
STFFDET1	2	11	1	0	0	Stiffness detection	[134]	S5
STFFDET2	2	15	1	0	0	Stiffness detection	[134]	S5
STR_FISH	2	18	1	0	0	Stream fish tracked by mark-recapture technique		S5
T_DIFFUS	5	45	1	3	0	Transdermal diffusion through two membranes with transition layer		E
TELEGRPH	4	16	2	0	0	Telegraph equation	[421]	S5
TIME_OPT	6	10	1	0	5	Time-optimal heat distribution	[424]	E
TONGUE	3	20	1	0	0	Motion of glacier tongue	[94]	S5
TRAFFIC	1	80	1	0	0	Traffic flow along a highway	[239]	S5
TRAN_DEG	3	32	1	0	0	Saturation of ground water (Richards equation)	[518], [400]	S1
TRANSDER	3	45	1	3	0	Transdermal diffusion		E
TRANSMEM	5	45	1	3	0	Two membranes with transition area		E
TRAV_WAV	2	140	1	0	0	Traveling waves (Burger's equation, exact solution known)	[3]	X
TUBE	1	63	3	0	0	Tube with gas separated by membrane, shock distribution	[503]	S1
TUBE0	2	10	1	0	0	Zero-order reaction in a catalytic-walled tube	[75]	S5
TWO_POPS	3	40	2	0	0	Two populations	[511]	S5
VISCOUS	5	5	5	5	0	Variable viscosities with periodic boundary	[12]	S5
WATER	1	50	2	0	0	Flow of shallow water over a barrier	[480], [228]	S1
WAVE1	1	180	2	0	0	Hyperbolic wave equation (exact solution known)	[420]	S0
WAVE2	2	100	2	0	0	Wave equation in form of two hyperbolic equations		X
WAVE3	2	60	2	0	0	Hyperbolic wave equation	[534]	S5
WAVE4	3	50	2	0	0	Two waves traveling in opposite directions, semi-hyperbolic sys-tem	[527], [433]	S1

10.7 Partial Differential Algebraic Equations

Again we proceed from r data sets

$$(t_i, y_i^k), \quad i = 1, \dots, l_t, \quad k = 1, \dots, r, \quad ,$$

where l_t time values and $l = l_t r$ corresponding measurement values are defined together with l weights w_i^k . Some of the weights can become zero in cases when the corresponding measurement value is missing, if artificial data are needed, or if plots are to be generated for state variables for which E data do not exist. The subsequent table contains the actual number $\tilde{l} \leq l$ of terms taken into account in the final least squares formulation.

The system of partial differential algebraic equations under consideration is

$$\begin{aligned} \dot{u}_1 &= F_1(p, u, u_x, u_{xx}, v, x, t) \quad , \\ &\dots \\ \dot{u}_{n_d} &= F_{n_d}(p, u, u_x, u_{xx}, v, x, t) \quad , \\ 0 &= F_{n_d+1}(p, u, u_x, u_{xx}, v, x, t) \quad , \\ &\dots \\ 0 &= F_{n_d+n_a}(p, u, u_x, u_{xx}, v, x, t) \quad , \end{aligned}$$

where $u_d = (u_1, \dots, u_{n_d})^T$ and $u_a = (u_{n_d+1}, \dots, u_{n_d+n_a})^T$ are the differential and algebraic state variables, $u = (u_d, u_a)^T$. $v \in \mathbb{R}^{n_c}$ denotes the state variables belonging to the coupled system of ordinary differential and algebraic equations. To simplify the notation, flux functions are omitted.

Initial and boundary conditions may depend on the parameter vector to be estimated. Since the starting time is assumed to be zero initial values have the form

$$u(p, x, 0) = u_0(p, x) \quad ,$$

where $u = (u_d, u_a)^T$ is the combined vector of all differential and algebraic state variables. For both end points x_L and x_R we allow Dirichlet or Neumann boundary conditions

$$\begin{aligned} u(p, x_L, t) &= u^L(p, v, t) \quad , \\ u(p, x_R, t) &= u^R(p, v, t) \quad , \\ u_x(p, x_L, t) &= \hat{u}^L(p, v, t) \quad , \\ u_x(p, x_R, t) &= \hat{u}^R(p, v, t) \end{aligned}$$

for $0 < t \leq T$, where T is the final integration time, for example the last E time value t_{l_t} . They may depend on the coupled ordinary differential and algebraic state variables. We do not require the evaluation of all boundary functions. Instead, we omit some of them depending on the structure of the PDAE model, for example, whether second partial derivatives

exist in the right-hand side or not. Moreover, arbitrary implicit boundary conditions can be formulated in form of coupled algebraic equations.

However, we must treat initial and boundary conditions with more care. We have to guarantee that at least the boundary and transition conditions satisfy the algebraic equations

$$\begin{aligned} 0 &= F_a(p, u(p, x_L, t), u_x(p, x_L, t), u_{xx}(p, x_L, t), v, x_L, t) , \\ 0 &= F_a(p, u(p, x_R, t), u_x(p, x_R, t), u_{xx}(p, x_R, t), v, x_R, t) . \end{aligned}$$

If initial conditions for discretized algebraic equations are violated, that is if equation

$$0 = F_a(p, u(p, x, 0), u_x(p, x, 0), u_{xx}(p, x, 0), v(p, 0), x, 0)$$

is inconsistent after inserting Dirichlet or Neumann boundary values and corresponding approximations for spatial derivatives, the corresponding system of nonlinear equations is solved internally proceeding from initial values given.

Each set of E data is assigned a spatial variable value $x_k \in [x_L, x_R]$, $k = 1, \dots, r$, where r denotes the total number of measurement sets. Some or all of the x_k -values may coincide, if different measurement sets are available at the same local position. Since partial differential equations are discretized by the method of lines, the fitting points x_k are rounded to the nearest line.

The resulting parameter estimation problems is

$$\begin{aligned} \min \quad & \sum_{k=1}^r \sum_{i=1}^{l_t} (w_i^k (h_k(p, u(p, x_k, t_i), u_x(p, x_k, t_i), \\ & \quad \quad \quad u_{xx}(p, x_k, t_i), v(p, t_i), t_i) - y_i^k))^2 \\ p \in \mathbb{R}^n : \quad & g_j(p) = 0 , \quad j = 1, \dots, m_e , \\ & g_j(p) \geq 0 , \quad j = m_e + 1, \dots, m_r , \\ & p_l \leq p \leq p_u , \end{aligned}$$

where $v(p, t)$ is the solution vector of an optional system of n_c coupled ordinary differential and algebraic equations similar to the previous section. It is assumed that all model functions $h_k(p, u, u_x, u_{xx}, v, t)$ and $g_j(p)$ are continuously differentiable subject to p for $k = 1, \dots, r$ and $j = 1, \dots, m_r$, and also the state variables and their spatial derivatives $u(p, x, t)$, $u_x(p, x, t)$, $u_{xx}(p, x, t)$, and $v(p, t)$.

Test problems with one-dimensional partial differential algebraic equations are listed in Table B.7. Not listed are the number of integration areas, switching times, and structure of the boundary conditions. There are no equality constraints.

Table B.7. Partial Differential Algebraic Equations

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_d</i>	<i>n_a</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
2ND_DIR1	3	20	1	1	0	0	Second order Dirichlet problem	[282]	S5
2ND_DIR2	3	20	1	1	0	0	Second order inhomogeneous Dirichlet problem	[282]	S5
ACCRET_A	1	20	6	4	0	0	Thermal equilibrium curves in Keplerian accretion disks	[381]	S5
ACCRET_F	1	112	3	1	0	0	Thermal equilibrium curves in Keplerian accretion disks	[381]	S5
ACETYL_T	2	180	10	9	0	0	Tubular acetylene reactor, time-dependent formulation	[44]	S0
ACETYL_Z	2	20	10	1	0	9	Tubular acetylene reactor, space-dependent formulation	[44]	S1
BEAM1	3	99	2	2	0	0	Curved beam	[520], [507]	S5
BEAM2	3	90	2	2	0	0	Linked beams	[520], [507]	S5
BIFURC1	3	180	2	0	2	0	Bifurcation with codimension 2 (Ginzburg-Landau equation)	[9]	S1
BIFURC2	2	300	2	0	2	0	Bifurcation with codimension 2 (Ginzburg-Landau equation), dense observation grid	[9]	S1
BVP_TRIV	1	10	1	1	0	0	Boundary value problem with known solution	[271]	X
CAPILL	2	13	1	1	0	0	Capillar filled with water under electric charge		S5
CHRESIGN	8	303	9	3	0	0	Transport effects for chromatographic resins		S5
CNT_CUR2	4	80	3	1	0	0	Counter-current separation of fluid phase concentrations with phase equilibrium	[379]	S5
CO_OXYD	3	68	2	0	2	0	CO oxydation on Pt(110)	[25]	S5
CTFLOW	1	40	4	2	0	0	Two incompressible counter-current flows of binary liquid mixture with semi-permeable wall	[339]	S1
CUSP	2	40	3	0	3	0	Threshold-nerve impulse with cusp catastrophe	[559]	S5
ELA_TUBE	3	40	2	1	0	0	Waves propagating in a liquid-filled elastic tube (Korteweg-de Vries-Burgers equation)	[241]	X
ELDYN_A	3	20	4	2	0	0	Electrodynamic application with algebraic equations	[50]	S5
EW_WAVE	2	48	2	1	0	0	Wave propagation in media with nonlinear steepening and dispersion	[201]	X

(continued)

<i>name</i>	<i>n</i>	<i>l</i>	<i>n_d</i>	<i>n_a</i>	<i>n_c</i>	<i>m_r</i>	<i>background</i>	<i>ref</i>	<i>data</i>
FLAT_MEM	1	20	1	0	2	0	Concentration boundary layer over a flat membrane		E
FLOWDIFF	2	101	2	0	2	0	Flow system with diffusion	[5]	S5
HEAT_A	1	27	2	1	0	0	Heat equation, formulated with algebraic equation		S5
HEAT_F	2	27	2	1	0	0	Heat equation, formulated with algebraic equation and flux formulation		S5
HEAT_NLC	2	10	1	0	1	0	Heat equation with nonlinear boundary condition of Stefan-Boltzmann type	[510]	E
HEAT_NLD	2	10	1	0	1	1	Heat equation with nonlinear boundary condition of Stefan-Boltzmann type	[510]	none
HEAT_R	2	27	2	1	0	9	Heat equation with dynamical restrictions and algebraic equation		S5
HEAT_RAD	2	10	2	1	0	0	Heat conduction with radiation and forced convection	[177]	S5
KDV1	2	44	2	1	0	0	Korteweg-de-Vries equation, exact solution with one soliton	[422]	X
KDVE	1	17	2	1	0	0	Shallow water flow, balancing front sharpening and dispersion to produce solitons	[422]	S0
MC_DIST	4	63	4	2	4	0	Multi-component distillation	[339]	S5
MCFC1	20	20	19	2	0	0	MCFC fuel cell	[215],[92]	none
MCFC2	3	110	19	2	0	0	MCFC fuel cell	[215],[92]	S1
MCFC3	3	110	19	2	0	0	MCFC fuel cell	[215],[92]	S1
MCMC_0	5	70	19	2	9	0	MCMC fuel cell		S1
NET_3	10	100	2	1	2	0	Network with three beams and controlled Neumann knot	[166]	E
PAR_SINA	2	84	2	1	0	0	Parabolic PDE with inhomogeneous sinus-term	[397], [380]	X
PDAE4	2	6	2	1	0	0	Simple fourth order PDAE with exact solution		X
PLASMA	3	20	4	2	0	0	Space-time movement of ions and electrons	[300]	S5
PRESSURE	3	15	3	1	0	0	Pressure-driven flow in porous media	[293]	S5
SILICON	1	102	2	1	0	0	Diffusion in silicon		S5
TUNNEL	2	24	2	1	0	0	Tunnel furnace with heating and cooling section	[339]	S1
UNLBEAM	2	10	5	3	0	0	Thin uniform cantilevered beam	[177]	S5
VIB_BEAM	6	6	3	1	0	0	Boundary control of transverse vibrations of a beam	[31]	E

Bibliography

- [1] Abbott M.B., Minns A.W. (1998): *Computational Hydraulics*, Ashgate, Aldershot
- [2] Abel O., Helbig A., Marquardt W. (1997): *Optimization approaches to control-integrated design of industrial batch reactors*, Technical Report LPT-1997-20, Lehrstuhl für Prozesstechnik, RWTH Aachen
- [3] Adjerid S., Flaherty J.E. (1986): *A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations*, SIAM Journal on Numerical Analysis, Vol. 23, 778-796
- [4] Ahmed N.U., Teo K.L. (1981): *Optimal Control of Distributed Parameter Systems*, Elsevier, Amsterdam
- [5] Alonso A.A., Banga J.R., Balsa-Canto J.R. (2002): *Model reduction of complex food processes with applications in control and optimization*, in: Computational Techniques in Food Engineering, Editorial CIMNE-UPC, Universidad Politecnica de Cataluna, Barcelona, ISBN 84-95999-13-7, 155-169
- [6] Anderson D.H. (1983): *Compartmental Modeling and Tracer Kinetics*, Lecture Notes in Biomathematics, Vol. 50, Springer, Berlin
- [7] Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. (1999): *LAPACK Users' Guide, Third Edition*, SIAM, Philadelphia
- [8] Andersson F., Olsson B. eds. (1985): *Lake Gårdsjön. An Acid Forest Lake and its Catchment*, Ecological Bulletins, Vol. 37, Stockholm
- [9] Argentine M., Coulet P. (1997): *Chaotic nucleation of metastable domains*, Physical Reviews E, Vol. 56, 2359-2362
- [10] Ascher U.M., Mattheij R., Russel R. (1995): *Numerical Solution of Boundary Value Problems*, SIAM, Philadelphia
- [11] Ascher U.M., Petzold L.R. (1998): *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia

- [12] Ascher U., Ruuth S., Wettin B. (1995): *Implicit-explicit methods for time-dependent partial differential equations*, SIAM Journal on Numerical Analysis, Vol. 32, 797-823
- [13] Atkinson A.C., Bogacka B. (2002): *Compound and other optimum designs for systems of non-linear differential equations arising in chemical kinetics*, Chemometrics and Intelligent Laboratory Systems, Vol. 61, 17 - 33
- [14] Baake E., Schloeder J.P. (1992): *Modelling the fast fluorescence rate of photosynthesis*, Bulletin of Mathematical Biology, Vol. 54, 999-1021
- [15] Baily J.E. , Ollis D.F. (1986): *Biochemical Engineering Fundamentals*, McGraw Hill, New York
- [16] Balsa-Canto E., Alonso A.A., Banga J.R. (2002): *A novel, efficient and reliable method for thermal process design and optimization. Part I: Theory*, Journal of Food Engineering, Vol. 52, 227-234
- [17] Balsa-Canto E., Banga J.R., Alonso A.A., Vassiliadis V.S. (1998): *Optimal control of distributed processes using restricted second order information*, Report, Chemical Engineering Lab., CSIC, University of Vigo, Spain
- [18] Balsa-Canto E., Banga J.R., Alonso A.A., Vassiliadis V.S. (2001): *Dynamic optimization of chemical and biochemical processes using restricted second-order information*, Computers and Chemical Engineering, Vol. 25, 539-546
- [19] Baltes M., Schneider R., Sturm C., Reuss M. (1994): *Optimal experimental design for parameter estimation in unstructured growth models*, Biotechnical Progress, Vol. 10, 480-488
- [20] Banga J.R., Singh R.P. (1994): *Optimisation of air drying of foods*, Journal of Food Engineering, Vol. 23, 189-221
- [21] Banga J.R., Alonso A.A., Singh R.P. (1997): *Stochastic dynamic optimization of batch and semicontinuous bioprocesses*, Biotechnology Progress, Vol. 13, 326-335
- [22] Banga J.R., Versyck K.J., Van Impe J.F. (2002): *Computation of optimal identification experiments for nonlinear dynamic process models: a stochastic global optimization approach*, Journal of Industrial Engineering and Chemical Research, Vol. 41, 2425-2430
- [23] Banks H.T., Crowley J.M., Kunisch K. (1983): *Cubic spline approximation techniques for parameter estimation in distributed systems*, IEEE Transactions on Automatic Control, Vol. AC-28, No. 7, 773-786
- [24] Banks H.T, Kunisch K. (1989): *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser, Boston, Basel, Berlin
- [25] Bär M., Hegger R., Kantz H (1999): *Fitting partial differential equations to space-time-dynamics*, Physical Reviews E, Vol. 59, 337-342

- [26] Bard Y. (1970): *Comparison of gradient methods for the solution of nonlinear parameter estimation problems*, SIAM Journal on Numerical Analysis, Vol. 7, 157-186
- [27] Bard Y. (1974): *Nonlinear Parameter Estimation*, Academic Press, New York, London
- [28] Bartholomew-Biggs M.C. (1995): *Implementing and using a FORTRAN 90 version of a subroutine for non-linear least squares calculations*, Report, NOC, Hatfield
- [29] Bauer I., Bock H.G., Körkel S., Schlöder J. (2000): *Numerical methods for optimum experimental design in DAE systems*, Journal of Computational and Applied Mathematics, Vol. 120, 1-25
- [30] Baumeister J. (1987): *Stable Solution of Inverse Problems*, Vieweg, Braunschweig
- [31] Bazeze A., Bruch J.C., Sloss J.M. (1999): *Numerical solution of the optimal boundary control of transverse vibrations of a beam*, Numerical Methods for Partial Differential Equations, Vol. 15, No. 5, 558-568
- [32] Beck J.V., Arnold K.J. (1977): *Parameter Estimation in Engineering and Science*, John Wiley, New York
- [33] Bellman R.E., Kalaba R.E., Lockett J. (1966): *Numerical Inversion of the Laplace Transform*, American Elsevier, New York
- [34] Belohlav Z., Zamostny P., Kluson P., Volf J. (1997): *Application of a random-search algorithm for regression analysis of catalytic hydrogenizations*, Canadian Journal of Chemical Engineering, Vol. 75, 735-742
- [35] Beltrami E. (1987): *Mathematics for Dynamic Modeling*, Academic Press, Orlando
- [36] Benecke C. (1993): *Interne numerische Differentiation von gewöhnlichen Differentialgleichungen*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [37] Berzins M., Dew P.M. (1991): *Algorithm 690: Chebyshev polynomial software for elliptic-parabolic systems of PDEs*, ACM Transactions on Mathematical Software, Vol. 17, No. 2, 178-206
- [38] Bethe H.A., Salpeter E.E. (1977): *Quantum Mechanics of One- and Two-Electron Atoms*, Plenum Press, New York
- [39] Bettenhausen D. (1996): *Automatische Struktursuche für Regler und Strecke*, Fortschrittsberichte VDI, Reihe 8, Nr. 474, VDI, Düsseldorf
- [40] Betts J.T. (1997): *Experience with a sparse nonlinear programming algorithm*, in: Large Scale Optimization with Applications, Part II: Optimal Design and Control, L.T. Biegler, T.F. Coleman, A.R. Conn, F.N. Santos eds., Springer, Berlin
- [41] Biegler L.T., Damiano J.J., Blau G.E. (1986): *Nonlinear parameter estimation: a case study comparison*, AIChE Journal, Vol. 32, No. 1, 29-45

- [42] Bird H.A., Milliken G.A. (1976): *Estimable functions in the nonlinear model*, Communications of Statistical Theory and Methods, Vol. 15, 513-540
- [43] Bird, R.B., Stewart W.E., Lightfoot E.N. (1960): *Transport Phenomena*, John Wiley, New York
- [44] Birk J., Liepelt M., Schittkowski K., Vogel F. (1999): *Computation of optimal feed rates and operation intervals for tubular reactors*, Journal of Process Control, Vol. 9, 325-336
- [45] Bischof C., Carle A., Corliss G., Griewank A., Hovland P. (1992): *ADIFOR: Generating derivative codes from Fortran programs*, Scientific Programming, Vol. 1, No. 1, 11-29
- [46] Bitterlich S., Knabner P. (2003): *Experimental design for outflow experiments based on a multi-level identification method for material laws*, Inverse Problems, Vol. 19, 1011-1030
- [47] Björck A. (1990): *Least Squares Methods*, Elsevier, Amsterdam
- [48] Black F., Scholes M. (1973): *The pricing of options and corporate liabilities*, Journal of Political Economics, Vol. 81, 637-659
- [49] Blatt M., Schittkowski K. (2000): *Optimal control of one-dimensional partial differential algebraic equations with applications*, Annals of Operations Research, Vol. 98, 45-64
- [50] Blom J.G., Zegeling P.A. (1994): *Algorithm 731: A moving grid interface for systems of one-dimensional time-dependent partial differential equations*, ACM Transactions on Mathematical Software, Vol. 20, No. 2, 194-214
- [51] Bock H.G. (1978): *Numerical solution of nonlinear multipoint boundary value problems with applications to optimal control*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 58, 407
- [52] Bock H.G. (1983): *Recent advantages in parameter identification techniques for ODE*, Proceedings of the International Workshop on Numerical Treatment of Inverse Problems in Differential and Integral Equations, Birkhäuser, Boston, Basel, Berlin 95-121
- [53] Bock H.G., Eich E., Schlöder J.P. (1987): *Numerical solution of constrained least squares problems in differential-algebraic equations*, Proceedings of the Fourth Seminar NUMDIFF-4, Halle, Numerical Treatment of Differential Equations, Teubner-Texte zur Mathematik, Vol. 104, Teubner, Stuttgart
- [54] Boderke P., Schittkowski K., Wolf M., Merkle H.P. (2000): *A mathematical model for diffusion and concurrent metabolism in metabolically active tissue*, Journal on Theoretical Biology, Vol. 204, No. 3, 393-407
- [55] Bojkor B., Hansel R., Luus R. (1993): *Application of direct search optimization to optimal control problems*, Hungarian Journal of Industrial Chemistry, Vol. 21, 177-185
- [56] Borggaard J., Burns J. (1997): *A PDE sensitivity method for optimal aerodynamic design*, Journal of Computational Physics, Vol. 136, No. 2, 366-384

- [57] Bossel H. (1992): *Modellbildung und Simulation*, Vieweg, Braunschweig
- [58] Box G.P., Hunter W.G., MacGregor J.F., Erjavec J. (1973): *Some problems associated with the analysis of multiresponse data*, Technometrics, Vol. 15, 33-51
- [59] Box G.P., Hunter W.G., Hunter J.S. (1978): *Statistics for Experimenters*, John Wiley, New York
- [60] Bryson A.E., Denham W.F., Dreyfus S.E. (1963): *Optimal programming problems with inequality constraints*, AIAA Journal, Vol. 1, No. 11, 2544-2550
- [61] Bryson A.E., Ho Y.C. (1975): *Applied Optimal Control*, Hemisphere, New York
- [62] Buchauer O., Hiltmann P., Kiehl M. (1992): *Sensitivity analysis of initial-value problems with applications to shooting techniques*, DFG-SPP-Report No. 403, Mathematisches Institut, TU München
- [63] Bulirsch R. (1971): *Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung*, Technical Report, Carl-Cranz-Gesellschaft, Oberpfaffenhofen
- [64] Bulirsch R., Kraft D. (1994): *Computational Optimal Control*, International Series of Numerical Mathematics, Vol. 111, Birkhäuser, Boston, Basel, Berlin
- [65] Butcher J.C. (1963): *Coefficients for the Study of Runge-Kutta integration processes*, Journal of the Australian Mathematical Society, Vol. 3, 185-201
- [66] Butcher J.C. (1964): *Integration processes based on Radau quadrature formulas*, Mathematics of Computations, Vol. 18, 233-244
- [67] Buwalda J.G., Ross G.J.S., Stribley D.B., Tinker P.B. (1982): *The development of endomycorrhizal root systems*, New Phytologist, Vol. 92, 391-399
- [68] Buzzi-Ferraris G., Facchi G., Forzetti P., Troncani E. (1984): *Control optimization of tubular catalytic decay*, Industrial Engineering in Chemistry, Vol. 23, 126-131
- [69] Buzzi-Ferraris G., Morbidelli M., Forzetti P., Carra S. (1984): *Deactivation of catalyst - mathematical models for the control and optimization of reactors*, International Chemical Engineering, Vol. 24, 441-451
- [70] Bykov V., Yablonskii G., Kim V. (1978): *On the simple model of kinetic self-oscillations in catalytic reaction of CO oxidation*, Doklady AN USSR, Vol. 242, 637-639
- [71] Byrne G.D., Hindmarsh A.C. (1987): *Stiff ODE solvers: A review of current and coming attractions*, Journal of Computational Physics, Vol. 70, 1-62
- [72] Caassen N., Barber S.A. (1976): *Simulation model for nutrient uptake from soil by a growing plant root system*, Agronomy Journal, Vol. 68, 961-964

- [73] Campbel J.H. (1976): *Pyrolysis of subbituminous coal as it relates to in situ gasification, Part 1: Gas evalution*, Report UCRL-52035, Lawrence Livermore Lab., Livermore, USA
- [74] Campbell S.L., Marszalek W. (1996): *The index of an infinite dimensional implicit system*, Mathematical Modelling of Systems, Vol. 1, No. 1, 1-25
- [75] Caracotsios M., Stewart W.E. (1985): *Sensitivity analysis of initial value problems with mixed ODE's and algebraic equations*, Computers and Chemical Engineering, Vol. 9, 359-365
- [76] Caracotsios M., Stewart W.E. (1995): *Sensitivity analysis of initial-boundary-value problems with mixed PDE's and algebraic equations*, Computers and Chemical Engineering, Vol. 19, 1019-1030
- [77] Carrasco E.F., Banga J.R. (1998): *A hybrid method for the optimal control of chemical processes*, Report, Chemical Engineering Lab., CSIC, University of Vigo, Spain
- [78] Carasso C., Raviart P.-A., Serre D. eds. (1986): *Nonlinear Hyperbolic Equations*, Lecture Notes in Mathematics, No. 1270, Springer, Berlin
- [79] Carasso C., Charrier P., Hanouzet B., Joly J.-L. (1989): *Nonlinear Hyperbolic Equations*, Lecture Notes in Mathematics, No. 1402, Springer, Berlin
- [80] Carver M.B. (1978): *Efficient integration over discontinuities in ordinary differential equation simulations*, Mathematics of Computer Simulations, Vol. 20, 190-196
- [81] Cash J.R., Karp A.H. (1990): *A variable order Runge-Kutta method for Initial values: problems with rapidly varying right-hand sides*, ACM Transactions on Mathematical Software, Vol. 16, No. 3, 201-222
- [82] Chakravarthy S.R., Osher S. (1984): *High resolution schemes and the entropy condition*, SIAM Journal on Numerical Analysis, Vol. 21, No. 5, 955-984
- [83] Chakravarthy S.R., Osher S. (1984): *Very high order accurate TVD schemes*, ICASE Report No. 84-44
- [84] Chakravarthy S.R., Osher S. (1985): *Computing with high-resolution upwind schemes for hyperbolic equations*, Lectures in Applied Mathematics, Vol. 22, 57-86, Springer, Berlin
- [85] Chang K.S. (1978): *Second-order computational methods for distributed parameter optimal control problems*, in: Distributed Parameter Systems, W.H. Ray, D.G. Lainiotis eds., Marcel Dekker, New York, Basel, 47-134
- [86] Chartres B.A., Stepleman R.S. (1976): *Convergence of linear multistep methods for differential equations with discontinuities*, Numerische Mathematik, Vol. 27, 1-10
- [87] Chemburkar R.M., Morbidelli M., Varma A. (1986): *Parametric sensitivity of a CSTR*, Chemical Engineering Science, Vol. 41, 1647
- [88] Chen J. (1991): *Abkühlungsvorgänge von Stahlplatten mit Spritzwasserbeaufschlagung*, Umformtechnische Schriften, Vol. 30

- [89] Chen G., Mills W.H. (1981): *Finite elements and terminal penalization for quadratic cost optimal control problems governed by ordinary differential equations*, SIAM Journal on Control and Optimization, Vol. 19, 744-764
- [90] Cherruault Y. (1986): *Explicit and numerical methods for finding optimal therapeutics*, Mathematical Modelling, Vol. 7, 173-183
- [91] Chicone C. (1999): *Ordinary Differential Equations with Applications*, Springer, New York
- [92] Chudej K., Petzet V., Scherdel S., Pesch H.J., Schittkowski K., Heidebrecht P., Sundmacher K. (2003): *Numerical simulation of a 1D model of a molten carbonate fuel cell*, PAMM-Proceedings of Applied Mathematics and Mechanics, Vol. 3, 521-522
- [93] Clark C. (1976): *Mathematical Bioeconomics*, Wiley-Intersciences, New York
- [94] Collatz L. (1960): *The Numerical Treatment of Differential Equations*, Springer, Berlin
- [95] Collin R.E. (1991): *Field Theory of Guided Waves*, IEEE Press, New York
- [96] Colombeau J.F., Le Roux (1986): *Numerical techniques in elastoplasticity*, in: Nonlinear Hyperbolic Problems, C. Carasso, P.-A. Raviart, D. Serre eds., Lecture Notes in Mathematics, No. 1270, Springer, Berlin
- [97] Crank J. (1970): *The Mathematics of Diffusion*, Oxford at the Clarendon Press
- [98] Cunge J.A., Holly F.M. (1980): *Practical Aspects of Computational River Hydraulics*, Pitman, Boston
- [99] Cuthrell J.E., Biegler L.T. (1989): *Simultaneous optimization nad solution methods for batch reactor control profiles*, Computational Chemical Engineering, Vol. 13, 49-62
- [100] Dahlquist G., Edsberg L., Skölleremo G., Söderlind G. (1982): *Are the numerical methods and software satisfactory for chemical kinetics?*, in: Numerical Integration of Differential Equations and Large Linear Systems, J. Hinze ed., Springer, Berlin
- [101] Daniel C., Wood F.S. (1980): *Fitting Equations to Data*, John Wiley, New York
- [102] Davidian M., Giltinan D.M. (1995): *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London
- [103] Davis H.T. (1962): *Introduction to Nonlinear Differential and Integral Equations*, Dover
- [104] de Saint-Venant, B. (1871): *Théorie du mouvement non-permanent des eaux avec application aux crues des rivières et à l'introduction des marées dans leur lit*, Comptes Rendus Academie des Sciences, Vol. 73, 148-154
- [105] Denbigh K.G. (1958): *Optimum temperature sequence in reactors*, Chemical Engineering Sciences, Vol. 8, 125-132

- [106] Dennis J.E.jr. (1973): *Some computational technique for the nonlinear least squares problem*, in: Numerical Solution of Systems of Nonlinear Algebraic Equations, G.D. Byrne, C.A. Hall eds., Academic Press, New York, London
- [107] Dennis J.E.jr. (1977): *Nonlinear least squares*, in: The State of the Art in Numerical Analysis, D. Jacobs ed., Academic Press, New York, London
- [108] Dennis J.E.jr., Gay D.M., Welsch R.E. (1981): *Algorithm 573: NL2SOL-An adaptive nonlinear least-squares algorithm*, ACM Transactions on Mathematical Software, Vol. 7, No. 3, 348-368
- [109] Dennis J.E.jr., Gay D.M., Welsch R.E. (1981): *Algorithm 573: NL2SOL-An adaptive nonlinear least-squares algorithm*, ACM Transactions on Mathematical Software, Vol. 7, No. 3, 369-383
- [110] Dennis J.E.jr., Heinkenschloss M., Vicente L.N. (1998): *Trust-region interior-point SQP algorithm for a class of nonlinear programming problems*, SIAM Journal on Control, Vol. 36, No. 5, 1750-1794
- [111] Deuffhard P. (1979): *A stepsize control for continuation methods with special applications to multiple shooting techniques*, Numerische Mathematik, Vol. 33, 115-146
- [112] Deuffhard P., Apostolescu V. (1977): *An underrelaxed Gauß-Newton method for equality constrained nonlinear least squares*, Proceedings of the IFIP Conference on Optimization Techniques, Part 2, A.V. Balakrishnan, Thoma M. eds., Lecture Notes in Control and Information Sciences, Vol. 7, 22-32, Springer, Berlin
- [113] Diener I. (1986): *Trajectory nets connecting all critical points of a smooth function*, Mathematical Programming, Vol. 36, 340-352
- [114] Dietrich E.E., Eigenberger G. (1996): *Compact finite difference methods for the solution of chemical engineering problems*, in: Scientific Computing in Chemical Engineering, Keil, Mackens, Voss, Werther eds., Springer, Berlin
- [115] Dobmann M., Liepelt M., Schittkowski K. (1995): *Algorithm 746: PCOMP: A Fortran code for automatic differentiation*, ACM Transactions on Mathematical Software, Vol. 21, No. 3, 233-266
- [116] Dobmann M., Liepelt M., Schittkowski K., Traßl C. (1995): *PCOMP: A Fortran code for automatic differentiation, language description and user's guide*, Report, Dept. of Mathematics, University of Bayreuth, Germany
- [117] Dobmann M., Schittkowski K. (1995): *PDEFIT: A Fortran code for constrained parameter estimation in partial differential equations, - user's guide -*, Report, Dept. of Mathematics, University of Bayreuth, Germany
- [118] Dolan E.D., Moré J. (2001): *Benchmarking optimization software with COPS*, Technical Report ANL/MCS-246, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, Illinois

- [119] Donaldson J.R., Schnabel R.B. (1987): *Computational experience with confidence regions and confidence intervals for nonlinear least squares*, Technometrics, Vol. 29, 67-82
- [120] Donat R., Marquina A. (1996): *Capturing shock reflections: An improved flux formula*, Journal on Computational Physics, Vol. 25, 42-58
- [121] Donea J., Huerta A. (2003): *Finite Element Methods for Flow Problems*, Wiley
- [122] Dormand J.R., Prince P.J. (1981): *High order embedded Runge-Kutta formulae*, Journal on Computational Applied Mathematics, Vol. 7, 67-75
- [123] Dorondicyn A.A. (1947): *Asymptotic solution of the van der Pol equation*, Prikl. Mat. i Meh., Vol. 11, 313-328, Translations AMS Ser. 1, Vol. 4, 1-23
- [124] Draper N.R., Smith H. (1981): *Applied Regression Analysis*, John Wiley, New York
- [125] DuChateau P. (1995): *An introduction to inverse problems in partial differential equations for engineers, physicists, and mathematicians, a tutorial*, in: Proceedings of the Workshop on Parameter Identification and Inverse Problems in Hydrology, Geology, and Ecology, J. Gottlieb, P. DuChateau eds., Kluwer Academic Publishers, Dordrecht, Boston, London 3 - 50
- [126] Dunn I.J., Heinzle E., Ingham J., Prenosil J.E. (1992): *Biological Reaction Engineering*, VCH, Weinheim
- [127] Dwyer H.A., Sanders B.R. (1978): *Numerical modeling of unsteady flame propagation*, Acta Astronautica, Vol. 5, 1171-1184
- [128] Edgar T.F., Himmelblau D.M. (1988): *Optimization of Chemical Processes*, McGraw Hill, New York
- [129] Edgar T.F., Lapidus L. (1972): *The computation of optimal singular bang-bang control II. Nonlinear systems*, AIChE Journal, Vol. 18, 780-785
- [130] Edsberg L., Wedin P.A. (1995): *Numerical tools for parameter estimation in ODE-systems*, Optimization Methods and Software, Vol. 6, 193-218
- [131] Ehrig R., Nowak U., Oeverdieck L., Deuffhard P. (1999): *Advanced extrapolation methods for large scale differential algebraic problems*, in: High Performance Scientific and Engineering Computing, H.-J. Bungartz, F. Durst, and Chr. Zenger (eds.), Lecture Notes in Computational Science and Engineering, Springer, Vol. 8, 233-244
- [132] Eich-Soellner E., Führer C. (1998): *Numerical Methods in Multibody Dynamics*, Teubner, Stuttgart
- [133] Eigenberger G., Butt J.B. (1976): *A modified Crank-Nicolson technique with non-equidistant space steps*, Chemical Engineering Sciences, Vol. 31, 681-691

- [134] Ekeland K., Owren B., Oines E. (1998): *Stiffness detection and estimation of dominant spectra with explicit Runge-Kutta methods*, ACM Transaction on Mathematical Software, Vol. 24, No. 4, 368-382
- [135] Elezgaray J., Arneodo A. (1992): *Crisis induced intermittent bursting in reaction-diffusion chemical systems*, Physical Reviews Letters, Vol. 68, 714-717
- [136] Ellison D. (1981): *Efficient automatic integration of ODEs with discontinuities*, Mathematics of Computational Simulations, Vol. 23, 12-20
- [137] Elnagar G.N., Kazemi M.A. (1998): *Pseudospectral Chebyshev optimal control of constrained nonlinear dynamical systems*, Computational Optimization and Applications, Vol. 11, No. 2, 195-213
- [138] Endrenyi, L. ed. (1981): *Kinetic Data Analysis*, Plenum Press, New York
- [139] Engleborghs K., Lust K., Roose D. (1999): *Direct computation of periodic doubling bifurcation points of large-scale systems of ODE's using a Newton-Picard method*, IMA Journal of Numerical Analysis, Vol. 19, 525-547
- [140] Engquist, B. (1986): *Computation of oscillatory solutions to partial differential equations*, in: C. Carasso, P.-A. Raviart, D. Serre eds., Nonlinear Hyperbolic Problems, Lecture Notes in Mathematics, No. 1270, Springer, Berlin
- [141] Enright W.H., Hull T.E. (1976): *Comparing numerical methods for the solution of stiff systems of ODEs arising in chemistry*, in: Numerical Methods for Differential Systems, L. Lapidus, W.E. Schiesser eds., Academic Press, New York, 45-66
- [142] Farnia K. (1976): *Computer-assisted experimental and analytical study of time/temperature-dependent thermal properties of the aluminium alloy 2024-T351*, Ph.D. Thesis, Dept. of Mechanical Engineering, Michigan State University
- [143] Fathi H., Schittkowski K., Hamielec A.E. (2004): *Dynamic modelling of living anionic solution polymerization of styrene/butadiene/divinyl benzene in a continuous stirred tank reactor train*, Polymer-Plastics Technology and Engineering Vol. 43, No. 3, 571-613
- [144] Fedkiw R.P., Merriman B., Donat R., Osher S. (1996): *The penultimate scheme for systems of conservation laws: Finite difference ENO with marquina's flux splitting*, UCLA CAM Report No. 96-18, Dept. of Mathematics, University of California at Los-Angeles
- [145] Feldman H.A. (1972): *Mathematical theory of complex ligand-binding systems at equilibrium: Some methods for parameter fitting*, Analytical Biochemistry, Vol. 48, 317-338
- [146] Fedorov V.V. (1972): *Theory of Optimal Experiments*, Academic Press, New York
- [147] Fermi E., Ulam S., Pasta J. (1974): *Studies of nonlinear problems I*, in: Nonlinear Wave Motion, Lectures on Applied Mathematics, AMS, Vol. 15, 143-155

- [148] Findeisen R., Allgöwer F. (2000): *A nonlinear model predictive control scheme for the stabilization of setpoint families*, Journal A, Benelux Quarterly Journal on Automatic Control, Vol. 41, 187-192
- [149] Fischer H. (1991): *Special problems in automatic differentiation*, in: Automatic Differentiation of Algorithms: Theory, Implementation and Application, A. Griewank, G. Corliss eds., SIAM, Philadelphia
- [150] Fischer P. (1996): *Modellierung und Simulation der Ammonium- und Nitrat-Dynamik in strukturierten Waldböden unter besonderer Berücksichtigung eines dynamischen, hierarchischen Wurzelsystems*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [151] Flaherty J.E., Moore P.K. (1995): *Integrated space-time adaptive hp-refinement methods for parabolic methods*, Applied Numerical Mathematics, Vol. 16, 317-341
- [152] Fogler H.S. (1974): *Elements of Chemical Kinetics and Reactor Calculations*, Prentice Hall, Englewood Cliffs, NJ
- [153] Fraley C. (1988): *Software performance on nonlinear least-squares problems*, Technical Report SOL 88-17, Dept. of Operations Research, Stanford University, Stanford, CA 94305-4022, USA
- [154] Frias J.M., Oliveira J.C, Schittkowski K. (2001): *Modelling of maltodextrin DE12 drying process in a convection oven*, Applied Mathematical Modelling, Vol. 24, 449-462
- [155] Friedman A., McLead B. (1986): *Blow-up of solutions of nonlinear degenerate parabolic equations*, Archive for Rational Mechanics and Analysis, Vol. 96, 55-80
- [156] Fu P.-C, Barford J.P. (1993): *Non-singular optimal control for fed-batch fermentation processes with a differential-algebraic system model*, Journal on Process Control, Vol. 3, No. 4, 211-218
- [157] Führer C. (1988): *Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen: Theorie, numerische Ansätze und Anwendungen*, Dissertation, Technical University of Munich
- [158] Führer C., Leimkuhler B. (1991): *Numerical solution of differential-algebraic equations for constrained mechanical motion*, Numerische Mathematik, Vol. 59, 55-69
- [159] Fujita H. (1975): *Foundations of Ultracentrifugical Analysis*, John Wiley, New York
- [160] Galer A.M., Crout N.M.J., Beresford N.A., Howard B.J., Mayes R.W., Barnett C.L., Eayres H., Lamb C.S. (1993): *Dynamic radiocaesium distribution in sheep: measurement and modelling*, Journal of Environmental Radiology, Vol. 20, 35-48
- [161] Gallant A.R. (1975): *Nonlinear Regression*, American Statistics, Vol. 29, No. 2, 73-81
- [162] Gallant A.R. (1987): *Nonlinear Statistical Models*, John Wiley, New York

- [163] Ganzha V.G., Vorozhtsov E.V. (1996): *Numerical Solutions for Partial Differential Equations*, CRC Press, Boca Raton, New York, London, Tokyo
- [164] Gear C.W. (1990): *Differential-algebraic equations, indices, and integral algebraic-equations*, SIAM Journal on Numerical Analysis, Vol. 27, 1527-1534
- [165] Gear C.W., Osterby O. (1984): *Solving ordinary differential equations with discontinuities*, ACM Transactions on Mathematical Software, Vol. 10, 23-44
- [166] Geisler J.(1999): *Dynamische Gebietszerlegung für Optimalsteuerungsprobleme auf vernetzten Gebieten unter Verwendung von Mehrgitterverfahren*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [167] Gelmi C., Perez-Correa R., Agosin E. (2003): *Modelling gibberella fujikuroi growth and GA3 production in solid state fermentation*, Report, Department of Chemical and Bioprocess Engineering, Pontificia Universidad Catolica de Chile, Casilla 306, Santiago 22, Chile
- [168] Gibaldi M., Perrier D. (1982): *Pharmacokinetics*, Marcel Dekker, New York, Basel
- [169] Gill P.E., Murray W. (1978): *Algorithms for the solution of the non-linear least-squares problem*, SIAM Journal on Numerical Analysis, Vol. 15, 977-992
- [170] Gill P.E., Murray W., Wright M.H. (1981): *Practical Optimization*, Academic Press, New York, London
- [171] Gill P.E., Murray W., Saunders M., Wright M.H. (1983): *User's Guide for SOL/NPSOL: A Fortran package for nonlinear programming*, Report SOL 83-12, Dept. of Operations Research, Stanford University, California
- [172] Gill P.E., Petzold L., Rosen J.B., Jay L.O., Park K. (1997): *Numerical optimal control of parabolic PDEs using DASOPT*, in: L. T. Biegler, T. F. Coleman, A. R. Conn and F. N. Santosa (eds.), Large Scale Optimization with Applications, Part II: Optimal Design and Control, IMA Volumes in Mathematics and its Applications, Volume 93, 271-300, Springer Verlag, Berlin, Heidelberg and New York
- [173] Godfrey K.R., DiStefano J.J. (1985): *Identifiability of model parameters*, in: IFAC Identification and System Parameter Estimation, P. Joungh ed., Pergamon Press, Oxford, 89-114
- [174] Goh C.J., Teo K.L. (1988): *Control parametrization: A unified approach to optimal control problems with general constraints*, Automatica, Vol. 24, 3-18
- [175] Gonzales-Concepcion C., Pestano-Gabino C. (1999): *Approximated solutions in rational form for systems of differential equations*, Numerical Algorithms, Vol. 21, 185-203
- [176] Goodman M.R. (1974): *Study Notes in System Dynamics*, Wright-Allen Press, Cambridge MA.
- [177] Goodson R.E., Polis M.P. (1978): *Identification of parameters in distributed systems*, in: Distributed Parameter Systems, W.H. Ray, D.G. Lainiotis eds., Marcel Dekker, New York, Basel, 47-134

- [178] Goodwin G.C., Payne R .L. (1977) *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, New York
- [179] Gorfine M., Freedman L., Shahaf G., Mehr R. (2003): *Maximum likelihood ratio test in complex models: An application to B lymphocyte development*, Bulletin of Mathematical Biology, Vol. 65, 1131-1139
- [180] Gottwald B.A., Wanner G. (1981): *A reliable Rosenbrock integrator for stiff differential equations*, Computing, Vol. 26, No.2, 355-360
- [181] Graf W.H. (1998): *Fluvial Hydraulics*, John Wiley, Chichester
- [182] Granvilliers L., Cruz J., Barahona P. (2000): *Parameter estimation using interval computations*, Report, Laboratoire d'Informatique, Universite de Nantes, B.P. 92208, F-44322 Nantes Cedex 3, France
- [183] Gray P, Scott S.K. (1990): *Chemical Oscillations and Instabilities*, Clarendon Press
- [184] Griewank A., Corliss G. (eds.) (1991): *Automatic Differentiation of Algorithms: Theory, Implementation and Application*, SIAM, Philadelphia
- [185] Griewank A., Juedes D., Srinivasan J. (1991): *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, Preprint MCS-P180-1190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, USA
- [186] Griewank A. (1989): *On automatic differentiation*, in: Mathematical Programming: Recent Developments and Applications, M. Iri, K. Tanabe eds., Kluwer Academic Publishers, Dordrecht, Boston, London, 83-107
- [187] Groch A.G. (1990): *Automatic control of laminar flow cooling in continuous and reversing hot strip mills*, Iron and Steel Engineer, 16-20
- [188] Gronwall T.H. (1919): *Note on the derivatives with respect to a parameter of the solutions of a system of differential equations*, Annals of Mathematics, Vol. 20, 292-296
- [189] Guckenheimer J., Holmes P. (1986): *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer, New York
- [190] Gugat M., Leugering G., Schittkowski K., Schmidt E.J.P.G. (2001): *Modelling, stabilization and control of flow in networks of open channels*, in: Online Optimization of Large Scale Systems, M. Grötschel, S.O. Krumke, J. Rambau eds., Springer, Berlin, 251-270
- [191] Gupta Y.P. (1995): *Bracketing method for on-line solution for low-dimensional nonlinear algebraic equations*, Industrial Engineering and Chemical Research, Vol. 34, 536-544
- [192] Guy R.H., Hadgraft J. (1988): *Physicochemical aspects of percutaneous penetration and its enhancement*, Pharmaceutical Research, Vol. 5, No. 12, 753-758
- [193] Haase G. (1990): *Dynamische Simulation einer Destillationskolonne und Entwurf einer Regelung*, Diploma Thesis, Berufsakademie Mannheim

- [194] Hadgraft J. (1979): *The epidermal reservoir, a theoretical approach*, International Journal of Pharmaceutics, Vol. 2, 265-274
- [195] Hahn H. (1921): *Theorie der reellen Funktionen*, Springer, Berlin
- [196] Hairer E., Lubich C., Roche M. (1989): *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Lecture Notes in Mathematics, Vol. 1409, Springer, Berlin
- [197] Hairer E., Nørsett S.P., Wanner G. (1993): *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series Computational Mathematics, Vol. 8, Springer, Berlin
- [198] Hairer E., Stoffer D. (1997): *Reversible long term integration with variable step sizes*, SIAM Journal on Scientific Computing, Vol. 10, 257-269
- [199] Hairer E., Wanner G. (1991): *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer Series Computational Mathematics, Vol. 14, Springer, Berlin
- [200] Hald J., Madsen K. (1981): *Combined LP and quasi-Newton methods for minmax optimization*, Mathematical Programming, Vol. 20, 49-62
- [201] Hamdi S., Gottlieb J.J., Hanson J.S. (2001): *Numerical solutions of the equal width wave equation using an adaptive method of lines*, in: Adaptive Methods of Lines, A. Vande Wouwer, Ph. Saucec Ph., W. Schiesser eds., Chapman and Hall/CRC, Boca Raton
- [202] Han S.-P. (1976): *Superlinearly convergent variable metric algorithms for general nonlinear programming problems* Mathematical Programming, Vol. 11, 263-282
- [203] Han S.-P. (1977): *A globally convergent method for nonlinear programming* Journal of Optimization Theory and Applications, Vol. 22, 297-309
- [204] Hanson R.J., Frogh F.T. (1992): *A quadratic-tensor model algorithm for nonlinear least-squares problems with linear constraints*, ACM Transactions on Mathematical Software, Vol. 18, No. 2, 115-133
- [205] Hao D.N., Reinhardt H.-J. (1998): *Gradient methods for inverse heat conduction problems*, in: Inverse Problems in Engineering, Vol. 6, No. 3, 177-211
- [206] Harten A., Engquist B., Osher S., Chakravarthy S.R. (1987): *Uniformly high order accurate essentially non-oscillatory schemes, III*, Journal on Computational Physics, Vol. 71, 231-303
- [207] Harten A. (1989): *ENO schemes with subcell resolution*, Journal on Computational Physics, Vol. 83, 148-184
- [208] Hartwanger C. (1996): *Optimierung von Antennenhörnern im Satellitenbau*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [209] Hartwanger C., Schittkowski K., Wolf H. (2000): *Computer aided optimal design of horn radiators for satellite communication*, Engineering Optimization, Vol. 33, 221-244

- [210] Haug E.J. (1989): *Computer-aided Kinematics and Dynamics of Mechanical Systems*, Allyn and Bacon
- [211] Hayashi H. (1989): *Drying technologies of foods-their history and future*, Drying Technology, Vol. 7, 315-369
- [212] Hayes B.T., Lefloch P.G. (1998): *Nonclassical shocks and kinetic relations: finite difference schemes*, SIAM Journal on Numerical Analysis, Vol. 35, No. 6, 2169-2194
- [213] Hearn A.C. (1978): *Reduce user's manual. Version 3.3*, Rand Publication CP78, Santa Monica, USA
- [214] Hedrich C. (1996): *Modellierung, Simulation und Parameterschätzung von Kühlprozessen in Walzstraßen*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [215] Heidebrecht P., Sundmacher K. (2003): *Molten carbonate fuel cell (MCFC) with internal reforming: Model-based analysis of cell dynamics*, Chemical Engineering Sciences, Vol. 58, 1029-1036
- [216] Heim A. (1998): *Modellierung, Simulation und optimale Bahnplanung von Industrierobotern*, Dissertation, Dept. of Mathematics, Technical University of Munich
- [217] Heinzl G., Woloszczak R., Thomann P. (1993): *TOPFIT 2.0: Pharmacokinetic and Pharmacodynamic Data Analysis System*, G. Fischer, Stuttgart, Jena, New York
- [218] Henninger R.J., Maudlin P.J., Rightly M.L. (1997): *Accuracy of differential sensitivities for one-dimensional shock problems*, Report LA-UR-97-2740, Los Alamos National Laboratory, Los Alamos, New Mexico 87545
- [219] Hilf K.-D. (1996): *Optimale Versuchsplanung zur dynamischen Roboterkalibrierung*, Fortschritt-Berichte VDI, Reihe 8, Nr. 590
- [220] Hines A.L., Maddox R.N. (1985): *Mass Transfer*, Prentice-Hall, Englewood-Cliffs
- [221] Hoch R. (1995): *Modellierung von Fließwegen und Verweilzeiten in einem Einzugsgebiet unter stationären Fließbedingungen*, Diploma Thesis, Fakultät of Biology, Chemistry and Geology, University of Bayreuth, Germany
- [222] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer, Berlin
- [223] Hohmann A. (1994): *Multilevel Newton h-p collocation*, ZIB Berlin, Preprint SC 94-25
- [224] Hooker P.F. (1965): *Benjamin Gompertz*, Journal of the Institute of Actuaries, Vol. 91, 203-212
- [225] Horbelt W., Timmer J., Melzer W. (1998): *Estimating parameters in nonlinear differential equations with application to physiological data*, Report, FDM, University of Freiburg

- [226] Horst R., Pardalos P.M. eds. (1995): *Handbook of Global Optimization*, Kluwe Academic Publishers, Dordrecht, Boston, London
- [227] Hotchkiss S.A.M. (1992): *Skin as a xenobiotic metabolizing organ*, in: *Process in Drug Metabolism*, G.G. Gibson ed., Taylor and Francis Ltd., London, 217-262
- [228] Houghton D.D., Kasahara A. (1968): *Nonlinear shallow flow over an isolated ridge*, *Communications of Pure and Applied Mathematics*, Vol. 21, 1-23
- [229] Hughes W.F., Brighton J.A. (1991): *Theory and Problems of Fluid Dynamics*, McGraw Hill, New York
- [230] Hull T.E., Enright W.H., Fellen B.M., Sedgwick A.E. (1972): *Comparing numerical methods for ordinary differential equations*, *SIAM Journal on Numerical Analysis*, Vol. 9, 603-637
- [231] Igler B., Knabner P. (1997): *Structural identification of nonlinear coefficient functions in transport processes through porous media*, Preprint No. 221, Dept. of Applied Mathematics, University of Erlangen, 1997
- [232] Igler B., Totsche K.U., Knabner P. (1997): *Unbiased identification of nonlinear sorption characteristics by soil column breakthrough experiments*, Preprint no. 224, Dept. of Applied Mathematics, University of Erlangen
- [233] Ihme F., Flaxa V. (1991): *Intensivkühlung von Fein- und Mittelstahl*, *Stahl und Eisen*, Vol. 112, 75-81
- [234] Ingham J., Dunn I.J., Heinzle E., Prenosil J.E. (1994): *Chemical Engineering Dynamics*, VCH, Weinheim
- [235] Jacobson D.H., Mayne D.Q. (1970): *Differential Dynamic Programming*, American Elsevier, New York
- [236] Jennings L.S., Fisher M.E., Teo K.L., Goh C.J. (1990): *MISER3 Optimal Control Software: Theory and User Manuel*, National Library of Australia
- [237] Jiang G.-S., Levy D., Lin C.-T., Osher S., Tadmor E. (1997): *High-resolution non-oscillatory schemes with non-staggared grids for hyperbolic conservation laws*, UCLA CAM Report 97-7, Dept. of Mathematics, University of California at Los-Angeles
- [238] Jiang G.-S., Shu C.-W. (1995): *Efficient implementation of weighted ENO-methods*, UCLA CAM Report 95-42, Dept. of Mathematics, University of California at Los-Angeles
- [239] Johnson C. (1998): *Adaptive finite element methods for conservation laws*, in: *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, B. Cockburn, C. Johnson, C.-W. Shu, E. Tadmor eds., *Lecture Notes in Mathematics*, Vol. 1697, Springer, Berlin
- [240] Johnson R.C., Jasik H. (1984): *Antenna Engineering*, McGraw Hill, New York
- [241] Johnson R.S. (1970): *A nonlinear equation incorporating damping and dispersion*, *Journal of Fluid Dynamics*, Vol. 42, 49-60

- [242] Jourdan, M. (1997): *Simulation und Parameteridentifikation von Destillationskolonnen*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [243] Juedes D.W. (1991): *A taxonomy of automatic differentiation tools*, in: Automatic Differentiation of Algorithms: Theory, Implementation and Application, A. Griewank, G. Corliss eds., SIAM, Philadelphia, 315-330
- [244] Kahaner D., Moler C., Nash S. (1989): *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs
- [245] Kalaba R., Spingarn K. (1982): *Control, Identification, and Input Optimization*, Plenum Press, New York, London
- [246] Kamke E. (1969): *Differentialgleichungen I*, Akademische Verlagsgesellschaft Geest und Portig
- [247] Kaps P., Rentrop P. (1979): *Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations*, Numerische Mathematik, Vol. 33, 55-68
- [248] Karlsen K.H., Lie K.-A. (1999): *An unconditionally stable splitting scheme for a class of nonlinear parabolic equations*, IMA Journal of Numerical Analysis, Vol. 19, 609-635
- [249] Kaps P., Poon S.W.H., Bui T.D. (1985): *Rosenbrock methods for stiff ODE's: A comparison of Richardson extrapolation and embedding techniques*, Computing, Vol. 34, No. 1, 17-40
- [250] Kelley C.T. (1999): *Iterative Methods for Optimization*, SIAM, Philadelphia
- [251] Kim I., Liebman M.J., Edgar T.F. (1990): *Robust error-in-variables estimation using nonlinear programming techniques*, AIChE Journal, Vol. 36, 985-996
- [252] Kim K.V. e.al. (1984): *An efficient algorithm for computing derivatives and extremal problems*, English translation, Ekonomika i matematicheskie metody, Vol. 20, No. 2, 309-318
- [253] Kletschkowski T., Schomburg U., Bertram A. (2001): *Viskoplastische Materialmodellierung am Beispiel des Dichtungswerkstoffes Polytetrafluorethylen*, Technische Mechanik, Vol. 3, 227-241
- [254] Knabner P., van Duijn C.J., Hengst S. (1995): *Crystal dissolution fronts in flows through porous media*, Report, Institute of Applied Mathematics, University of Erlangen
- [255] Ko D.Y.C., Stevens W.F. (1971): *Study of singular solutions in dynamic optimization*, AIChE Journal, Vol. 17, 160-166
- [256] Kojouharov, Chen B.M. (1999): *Nonstandard methods for the convective-dispersive transport equation with nonlinear reactions*, Numerical Methods for Partial Differential Equations, Vol. 16, No. 1, 107-132
- [257] Kopp R., Philipp F.D. (1992): *Physical parameters and boundary conditions for the numerical simulation of hot forming processes*, Steel Research, Vol. 63, 392-398

- [258] Kowalik J. (1967): *A note on nonlinear regression analysis*, Australian Computational Journal, Vol. 1, 51-53
- [259] Kripfganz J., Perl H. (1994): *Arbeiten mit Mathematica*, Carl Hanser, Oldenburg
- [260] Kuhn U., Schmidt G. (1987): *Fresh look into the design and computation of optimal output feedback controls for linear multivariable systems*, International Journal on Control, Vol. 46, No. 1, 75-95
- [261] Kühn E., Hombach V. (1983): *Computer-aided analysis of corrugated horns with axial or ring-loaded radial slots*, Report, Research Institute of the Deutsche Bundespost, Germany
- [262] Kung A.H.C., Baugham R.A., Larrick J.W. (1993): *Therapeutic Proteins*, W.H. Freeman, New York
- [263] Kuzmic P. (1998): *Fixed-point methods for computing the equilibrium composition of complex biochemical mixtures*, Biochemical Journal, Vol. 331, 571-575
- [264] Kuzmic P. (1999): *General numerical treatment of competitive binding kinetics: Application to thrombin-dehydrothrombin-hirudin*, Analytical Biochemistry, Vol. 267, 17-23
- [265] Kuznetsov, V.A., Puri R.K. (1999): *Kinetic analysis of high-affinity forms of interleukin-13 receptors*, Biophysical Journal, Vol. 77, 154-172
- [266] Lafon F., Osher S. (1991): *High order filtering methods for approximating hyperbolic systems of conservation laws*, Journal on Computational Physics, Vol. 96, 110-142
- [267] Lambert J.D. (1991): *Numerical Methods for Ordinary Differential Systems: The Initial-Value Problem*, John Wiley, New York
- [268] Lanczos C. (1956): *Applied Analysis*, Prentice Hall, Englewood Cliffs
- [269] Lagugne-Labarthe F., Bruneel J.L., Sourisseau C., Huber M.R., Börger V., Menzel H. (2002): *A microspectrometric study of the azobenzene chromophore orientation in a holographic diffraction grating inscribed on a p(HEMA-co-MMA) functionalized copolymer film*, Journal of Raman Spectroscopy, Vol. 32, 665-675
- [270] Lang J. (1993): *KARDOS: Cascade reaction diffusion one-dimensional system*, Technical Report TR 93-9, ZIB Berlin
- [271] Langtangen H.P. (1999): *Computational Partial Differential Equations*, Lecture Notes in Computational Science and Engineering, Vol. 2, Springer, Berlin, Heidelberg
- [272] Lapidus, L., Luus, R. (1967): *Optimal Control of Engineering Processes*, Blaisdell, Waltham, Mass.
- [273] Lapidus L., Aiken R.C., Liu Y.A. (1973): *The occurrence and numerical solution of physical and chemical systems having widely varying time constants*, in: *Stiff Differential Systems*, E.A. Willoughby ed., Plenum Press, New York, 187-200

- [274] Lastman G.J., Wentzell R.A., Hindmarsh A.C. (1978): *Numerical solution of a bubble cavitation problem*, Journal of Computational Physics, Vol. 28, 56-64
- [275] Lecar M. (1968): *Comparison of eleven numerical integrations of the same gravitational 25-body problem*, Bulletin Astronomique, Vol. 3, 91
- [276] Lee J., Ramirez W.F. (1994): *Optimal fed-batch control of induced foreign protein production by recombinant bacteria*, AIChE Journal, Vol. 40, 899-907
- [277] Lee T.T., Wang F.Y., Newell R.B. (1999): *Dynamic modelling and simulation of a complex biological process based on distributed parameter approach*, AIChE Journal, Vol. 45, No. 10, 2245-2268
- [278] Lefever R., Nicolis G. (1971): *Chemical instabilities and sustained oscillations*, Journal of Theoretical Biology, Vol. 30, 267-284
- [279] Leis J.E., Kramer M.A. (1988): *The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations*, ACM Transactions on Mathematical Software, Vol. 14, No. 1, 45-60
- [280] Leis J.E., Kramer M.A. (1988): *Algorithm 658: ODESSA - An ordinary differential equation solver with explicit simultaneous sensitivity analysis*, ACM Transactions on Mathematical Software, Vol. 14, No. 2, 61-67
- [281] Levenberg K. (1944): *A method for the solution of certain problems in least squares*, Quarterly Applied Mathematics, Vol. 2, 164-168
- [282] Lewis R.M., Patera A.T., Peraire J. (2000): *A posteriori finite element bounds for sensitivity derivatives of partial-differential-equation outputs*, Finite Elements in Design, Vol. 34, 271-290
- [283] Lewis J.W. (1994): *Modeling Engineering Systems*, LLH Technology Publishing, 1994
- [284] Liepelt M., Schittkowski K. (2000): *Algorithm 746: New features of PCOMP, a FORTRAN code for automatic differentiation*, ACM Transactions on Mathematical Software, Vol. 26, No. 3, 352-362
- [285] Liepelt M., Schittkowski K. (2000): *Optimal Control of Distributed Systems with Break Points*, in: Online Optimization of Large Scale Systems, M. Grötschel, S.O. Krumke, J. Rambau eds., Springer, Berlin, 271-294
- [286] Lindberg P.O., Wolf A. (1998): *Optimization of the short term operation of a cascade of hydro power stations*, in: Optimal Control: Theory, Algorithms, and Applications, W.W. Hager, P.M. Padalos eds., Kluwer Academic Publishers, Dordrecht, Boston, London, 326-345
- [287] Lindström P. (1982): *A stabilized Gauß-Newton algorithm for unconstrained least squares problems*, Report UMINF-102.82, Institute of Information Processing, University of Umea, Umea, Sweden

- [288] Lindström P. (1983): *A general purpose algorithm for nonlinear least squares problems with nonlinear constraints*, Report UMINF-103.83, Institute of Information Processing, University of Umea, Umea, Sweden
- [289] Lioen W.M., de Swart J.J.B. (1999): *Test set for initial value solvers, Release 2.1*, CWI, Amsterdam, The Netherlands
- [290] Liska R., Wendroff B. (1998): *Composite schemes for conservation laws*, SIAM Journal on Numerical Analysis, Vol. 35, No. 6, 2250-2271
- [291] Liu X.-D., Osher S. (1997): *Convex ENO high order multi-dimensional schemes without field by field decomposition or staggered grids*, UCLA CAM Report 97-26, Dept. of Mathematics, University of California at Los Angeles
- [292] Logan J.D. (1994): *An Introduction to Nonlinear Partial Differential Equations*, John Wiley, New York
- [293] Logan J.M. (2001): *Transport Modeling in Hydrochemical Systems*, Interdisciplinary Applied Mathematics, Springer, New York
- [294] Lohmann T. (1988): *Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, Dissertation, Dept. of Mathematics, University of Bonn
- [295] Lohmann T.W. (1997): *Modellierung und Identifizierung der Reaktionskinetik der Kohlepyrolyse*, Fortschrittsberichte VDI, Reihe 3, No. 499, VDI, Düsseldorf
- [296] Lohmann T.W., Bock H.G., Schlöder J.P. (1992): *Numerical methods for parameter estimation and optimal experiment design in chemical reaction systems*, Industrial and Engineering Chemistry Research, Vol. 31, 54-57
- [297] Lorenz, E.N. (1963): *Deterministic nonperiodic flow*, Journal of Atmospheric Sciences, Vol. 20, 130-141
- [298] Loth H., Schreiner T., Wolf M., Schittkowski K., Schäfer U. (2001): *Fitting drug dissolution measurements of immediate release solid dosage forms by numerical solution of differential equations*, submitted for publication
- [299] Lubich C. (1993): *Integration of stiff mechanical systems by Runge-Kutta methods*, ZAMP, Vol. 44, 1022-1053
- [300] Lucht W., Debrabant K. (1996): *Models of quasi-linear PDAEs with convection*, Report, Dept. of Mathematics and Computer Science, University of Halle, Germany
- [301] Lucht W., Strehmel K. (1998): *Discretization based indices for semilinear partial differential algebraic equations*, Applied Numerical Mathematics, Vol. 28, 371-386
- [302] Luenberger D.G. (1979): *Introduction to Dynamic Systems - Theory, Models, and Applications*, John Wiley, New York

- [303] Luksan L. (1985): *An implementation of recursive quadratic programming variable metric methods for linearly constrained nonlinear minmax approximations*, Kybernetika, Vol 21, 22-40
- [304] Luus R. (1974): *Two-pass method for handling difficult equality constraints in optimization*, AIChE Journal, Vol. 20, 608-610
- [305] Luus, R. (1993): *Optimization of fed-batch fermentors by iterative dynamic programming*, Biotechnology and Bioengineering, Vol. 41, 599-602
- [306] Luus R. (1993): *Optimal control of batch reactors by iterative dynamic programming*, Journal of Process Control, Vol. 4, No. 4, 218-226
- [307] Luus R. (1998): *Iterative dynamic programming: From curiosity to a practical optimization procedure*, Control and Intelligent Systems, Vol. 26, No. 1, 1-8
- [308] Luus R. (2000): *Iterative Dynamic Programming*, Chapman and Hall/CRC, Boca Raton, London, New York, Washington
- [309] Luus R. (2002): *Global optimization of Yeos optimal control problem*, Proceedings of the IASTED Conference, Cancun, Mexico, 71-74
- [310] Luyben W.L. (1973): *Process Modeling: Simulation and Control for Chemical Engineers*, McGraw Hill, New York
- [311] Luyben W.L. (1990): *Process Modeling: Simulation and Control for Chemical Engineers*, McGraw Hill, New York
- [312] Machielsen K.C.P. (1987): *Numerical solution of optimal control problems with state constraints by sequential quadratic programming in function space*, CWI Tract, Amsterdam
- [313] Madsen N.K., Sincovec R.F. (1976): *Software for partial differential equations*, in: Numerical Methods for Differential Systems, L. Lapidus, W.E. Schiesser eds., Academic Press, New York
- [314] Mahdavi-Amiri N. (1981): *Generally constrained nonlinear least squares and generating nonlinear programming test problems: Algorithmic approach*, Dissertation, The John Hopkins University, Baltimore, Maryland, USA
- [315] Majer C. (1998): *Parameterschätzung, Versuchsplanung und Trajektorienoptimierung für verfahrenstechnische Prozesse*, Fortschrittberichte VDI, Reihe 3, Nr. 538, VDI, Düsseldorf
- [316] Majer C., Marquardt W., Gilles E.D. (1995): *Reinitialization of DAE's after discontinuities*, Proceedings of the Fifth European Symposium on Computer-Aided Process Engineering, 507-512
- [317] Mannshardt R. (1978): *One-step methods of any order for ordinary differential equations with discontinuous right hand sides*, Numerische Mathematik, Vol. 31, 131-152
- [318] Maria G. (1989): *An adaptive strategy for solving kinetic model concomitant estimation-reduction problems*, Canadian Journal of Chemical Engineering, Vol. 67, 825-837

- [319] Marion M., Mollard A. (1999): *A multilevel characteristics method for periodic convection-dominated diffusion problems*, Numerical Methods for Partial Differential Equations, Vol. 16, No. 1, 107-132
- [320] Marquardt D. (1963): *An algorithm for least-squares estimation of nonlinear parameters*, SIAM Journal of Applied Mathematics, Vol. 11, 431-441
- [321] Marquina A., Donat R. (1993): *Capturing shock reflections: A nonlinear local characteristic approach*, UCLA CAM Report No. 93-31, Dept. of Mathematics, University of California at Los-Angeles
- [322] Marquina A., Osher S. (2000): *Explicit algorithms for a new time-dependent model based on level set motion for nonlinear deblurring and noise removal*, Report, Dept. of Mathematics, University of California, Los Angeles
- [323] Martinson W.S., Barton P.I. (1996): *A differentiation index for partial differential equations*, SIAM Journal on Scientific Computing, Vol. 21, No. 6, 2295-2315
- [324] Mattheij R., Molenaar J. (1996): *Ordinary Differential Equations in Theory and Practice*, John Wiley, Chichester, UK
- [325] Mattie H., Zhang L.-C., van Strijen E., Razab Sekh B., Douwes-Idema A.E.A. (1997): *Pharmacokinetic and pharmacodynamic models of the antistaphylococcal effects of Meropenem and Cloxacillin in vitro and in experimental infection*, Antimicrobial Agents and Chemotherapy, Vol. 41, No. 10, 2083-2088
- [326] Maurer H., Weigand M. (1992): *Numerical solution of a drug displacement problem with bounded state variables*, Optimal Control Applications and Methods, Vol. 13, 43-55
- [327] Mayer U. (1993): *Untersuchungen zur Anwendung eines Einschnitt-Polynom-Verfahrens zur Integration von Differentialgleichungen und DA-Systemen*, Ph.D. Thesis, Dept. of Chemical Engineering, University of Stuttgart
- [328] Mayr L.M., Odefey C., Schutkowski M., Schmid F.X. (1996): *Kinetic analysis of the unfolding and refolding of ribonuclease T1 by a stopped-flow double-mixing technique*, Biochemistry, Vol. 35, 5550-5561
- [329] Meadows D.H., Meadows D.L., Randers J. (1992): *Beyond the Limits*, Chelsea Green, Post Mills
- [330] Meissner E. (2000): *Messung von kurzen Konzentrationsprofilen mit Hilfe der analytischen TEM-EDX am Beispiel der Bestimmung von Diffusionskoeffizienten für Mg-Fe Interdiffusion in Olivin*, Dissertation, Faculty of Biology, Chemistry, and Geological Sciences, University of Bayreuth
- [331] Miele A., Wang T., Melvin W.W. (1987): *Optimal abort landing trajectories in the presence of windshear*, Journal of Optimization Theory and Applications, Vol. 12, 815-821

- [332] Mishkin M.A., Saguy I., Karel M. (1982): *Applications of optimisation in food dehydration*, Food Technology, Vol. 36, 101-109
- [333] Mishkin M.A. (1983): *Dynamic modeling, simulation and optimization of quality changes in air-drying of foodstuffs*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA
- [334] Mishkin M.A. (1983): *Dynamic optimization of dehydration processes: Minimizing browning in dehydration of potatoes*, Journal of Food Science, Vol. 48, 1617-1621
- [335] Missel P.J. (2000): *Finite element modeling of diffusion and partitioning in biological systems*, Report, Drug Delivery, Alcon Research Ltd., Fort Worth, USA
- [336] Mittelman H.D. (2001): *Sufficient optimality for discretized parabolic and elliptic control problems*, in: Fast Solution of Discretized Optimization Problems, K.-H. Hoffmann, R.H.W. Hoppe, and V. Schulz (eds.), ISNM 138, Birkhäuser, Basel
- [337] Mittra R. (1973): *Computer Techniques for Electromagnetics*, Pergamon Press, Oxford
- [338] Mohler R.R., Farooqi Z., Heilig T. (1984): *An immune lymphocyte circulation system*, in: System Modelling and Optimization, P. Thoft-Christensen ed., Lecture Notes in Control and Information Sciences, Vol. 59, Springer, 694-702
- [339] Molander M. (1990): *Computer aided modelling of distributed parameter process*, Technical Report No. 193, School of Electrical and Computer Engineering, Chalmers University of Technology, Göteborg, Sweden
- [340] Moré J.J. (1977): *The Levenberg-Marquardt algorithm: implementation and theory*, in: Numerical Analysis, G. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer, Berlin
- [341] Moré J.J., Garbow B.S., Hillstom K.E. (1981): *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, Vol. 7, No. 1, 17-41
- [342] Morrison R.T., Boyd R.N. (1983): *Organic Chemistry*, Allyn and Bacon
- [343] Morton K.W., Mayers D.F. (1994): *Numerical Solution of Partial Differential Equations*, Cambridge University Press
- [344] Munack A. (1995): *Simulation bioverfahrenstechnischer Prozesse*, in: Prozesssimulation, H. Schuler ed., VCH, Weinheim, 409-455
- [345] Munack A., Posten C. (1989): *Design of optimal dynamical experiments for parameter estimation*, Proceedings of the American Control Conference, Vol. 4, 2010-2016
- [346] Müller T.G., Noykova N., Gyllenberg M., Timmer J. (2002): *Parameter identification in dynamical models of anaerobic wastewater treatment*, Mathematical Biosciences, Vol. 177/178, 147-160
- [347] Murray, J.D. (1990): *Mathematical Biology*, Springer, New York

- [348] Naguma J., Arimoto S., Yoshizawa (1962): *An active pulse transmission line simulating nerve axon*, Proceedings of the IRE, Vol. 50, 2061-2070
- [349] Nagurka M.L. (1990): *Fourier-based optimal control of nonlinear dynamic systems*, Journal on Dynamical Systems, Measurements and Control, Vol. 112, 17-26
- [350] Nayfeh A. (1972): *Perturbation Analysis*, John Wiley, New York
- [351] Neittaanmäki P., Tiba D. (1994): *Optimal Control of Nonlinear Parabolic Systems*, Marcel Dekker, New York, Basel
- [352] Nelder J.A., Mead R. (1965): *A simplex method for function minimization*, The Computer Journal, Vol. 7, 308
- [353] Nelson K.A. (1993): *Using the glass transition approach for understanding chemical reaction rates in model food systems*, Ph.D. Thesis, Minnesota University, USA
- [354] Nelson K.A., Labuza T.P. (1994): *Water activity and food polymer science: implications of state on Arrhenius and WLF models in predicting shelf life*, Journal of Food Engineering, Vol. 22, 271-289
- [355] Nelson W. (1981): *Analysis of performance-degradation data*, IEEE Transactions on Reliability, Vol. 2, No. 2, 149-155
- [356] Newman P.A., Hou G.J.W., Taylor A.C. (1996): *Observations regarding use of advanced CFD analysis, sensitivity analysis, and design codes in MDO*, ICASE Report No. 96-16, NASA Langley Research Center, Hampton, Virginia 23681
- [357] Newell R.B., Lee P.L. (1989): *Applied Process Control - A Case Study*, Prentice Hall, Englewood Cliffs, New Jersey
- [358] Nickel B. (1995): *Parameterschätzung basierend auf der Levenberg-Marquardt-Methode in Kombination mit direkter Suche*, Diploma Thesis, Dept. of Mathematics, University of Bayreuth, Germany
- [359] Nishida N., Ichikawa A., Tazaki E. (1972), *Optimal design and control in a class of distributed parameter systems under uncertainty*, AIChE Journal, Vol. 18, 561-568
- [360] Nocedal J., Wright J. (1999): *Numerical Optimization*, Springer Series in Operational Research, Springer, New York
- [361] Nowak U. (1995): *A fully adaptive MOL-treatment of parabolic 1D-problems with extrapolation techniques*, Preprint SC 95-25, ZIB Berlin
- [362] Noykova N., Müller T.G., Gyllenberg M., Timmer J. (2001): *Quantitative analysis of anaerobic wastewater treatment processes: Identifiability and parameter estimation*, Biotechnology and Bioengineering, Vol. 78, 89-103
- [363] Oberle H.J. (1987): *Numerical Computation of Singular Control Functions for a Two-Link Robot Arm*, Lecture Notes in Control and Information Sciences, Vol. 95, Springer, Berlin

- [364] Odefey C., Mayr L.M., Schmid F.X. (1995): *Non-prolyl cis-trans peptide bond isomerization as a rate-determining step in protein unfolding and refolding*, Journal of Molecular Biology, Vol. 245, 69-78
- [365] Ogden R.W., Saccomandi G., Sgura I. (2004): *Fitting hyperelastic models to experimental data*, Computational Mechanics, Vol. 34, 484-502
- [366] Oh S.H., Luus R. (1975): *Optimal feedback control of time-delay systems*, AIChE Journal, Vol. 22, 144-147
- [367] Olansky A.S., Deming S.N. (1976): *Optimization and interpretation of absorbance response in the determination of formaldehyde with chromotropic acid*, Analytica Chimica Acta, Vol. 83, 241-249
- [368] Osborne M.R. (1972): *Some aspects of nonlinear least squares calculations*, in: Numerical Methods for Nonlinear Optimization, F. Lootsma ed., Academic Press, New York
- [369] Otey G.R., Dwyer H.A. (1979): *Numerical study of the interaction of fast chemistry and diffusion*, AIAA Journal, Vol. 17, 606-613
- [370] Otter M., Türk S. (1988): *The DFVLR models 1 and 2 of the Manutec R3 robot*, DFVLR-Mitteilungen 88-3, DFVLR, Oberpfaffenhofen, Germany
- [371] Ou L.-T. (1985): *2,4-D degradation and 2,4-D degrading microorganisms in soils*, Soil Sciences, Vol. 137, 100-107
- [372] Pantelides C.C., Gritsis D., Morison K.R., Sargent R.W.H. (1988): *The mathematical modeling of transient systems using differential-algebraic equations*, Computers and Chemical Engineering, Vol. 12, 440-454
- [373] Papalambros P.Y., Wilde D.J. (1988): *Principles of Optimal Design*, Cambridge University Press
- [374] Park S., Ramirez W.F. (1988): *Optimal production of secreted protein in fed-batch reactors*, AIChE Journal, Vol. 34, No. 8, 1550-1558
- [375] Peano G. (1890): *Démonstration de l'intégrabilité des équations différentielle ordinaires*, Mathematische Annalen, Vol. 37, 182-228
- [376] Pennington S.V., Berzins M. (1994): *New NAG Library software for first-order partial differential equations*, ACM Transactions on Mathematical Software, Vol. 20, No. 1, 63-99
- [377] Peters N., Warnatz J. eds. (1982): *Numerical Methods in Laminar Flame Propagation*, Notes on Numerical Fluid Dynamics, Vol. 6, Vieweg, Braunschweig
- [378] Petzold L.R. (1982): *A description of DASSL: A differential/algebraic system solver*, in: Proceedings of the 10th IMACS World Congress, Montreal, Canada
- [379] Pfeiffer B.-M., Marquardt W. (1996): *Symbolic semi-discretization of partial differential equation systems*, Mathematics and Computers in Simulation, Vol. 42, 617-628

- [380] Pfeleiderer J., Reiter J. (1991): *Biplicit numerical integration of partial differential equations with the transversal method of lines*, Report No. 279, DFG SPP *Anwendungsbezogene Optimierung und Steuerung*, Technical University, Dept. of Mathematics, Munich
- [381] Pin-Gao Gu, E.T. Vishniac, J.K. Cannizo (2000): *Thermal equilibrium curves and turbulent mixing in Keplerian accretion disks*, The Astrophysical Journal, Vol. 534, 380-397
- [382] Pinter J.D. (1995): *Global Optimization in Action*, Kluwer Academic Publishers, Dordrecht, Boston, London
- [383] Plusquellec Y., Courbon F., Nogarede S., Houin G. (1998): *Consequence of equal absorption, distribution and/or elimination rate constants*, Report, UFR de Mathematiques, Universite Paul Sabatier, Toulouse
- [384] Poepe C., Pellicari C., Bachmann K. (1979): *Computer analysis of Feulgen hydrolysis kinetics*, Histochemistry, Vol. 60, 53-60
- [385] Pohjanpalo H. (1978): *System identifiability based on power series expansion of the solution*, Mathematical Bioscience, Vol. 41, 21-33
- [386] Posten C., Munack A. (1989): *On-line application of parameter estimation accuracy to biotechnical processes*, Proceedings of the Americal Control Conference, Vol. 3, 2181-2186
- [387] Powell M.J.D. (1978): *A fast algorithm for nonlinearly constraint optimization calculations*, in: Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer, Berlin
- [388] Powell M.J.D. (1978): *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in: Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds., Academic Press, New York, London
- [389] Pratt W.B., Taylor P. (1990): *Principles of Drug Action*, Churchill Livingstone, New York
- [390] Preston A.J., Berzins M. (1991): *Algorithms for the location of discontinuities in dynamic simulation problems*, Computers and Chemical Engineering, Vol. 15, 701-713
- [391] Price H., Varga R., Warren J. (1966): *Application of oscillation matrices to diffusion-convection equations*, Journal of Methematical Physics, 301-311
- [392] Räumschüssel S. (1998): *Rechnerunterstützte Vorverarbeitung und Codierung verfahrenstechnischer Modelle für die Simulationsumgebung DIVA*, Fortschrittberichte VDI, Reihe 20, Nr. 270, VDI, Düsseldorf
- [393] Radu F.A., Bause M., Knabner P., Lee G.W., Friess W.C. (2001): *Drug release from collagen matrices*, Preprint IAM Erlangen Nr. 284, University of Erlangen, D- 91058 Erlangen
- [394] Ramsin H., Wedin P.A. (1977): *A comparison of some algorithms for the nonlinear least squares problem*, Nordisk Tidstr. Informationsbehandling (BIT), Vol. 17, 72-90

- [395] Ratkowsky D.A. (1988): *Nonlinear Regression Modeling*, Marcel Dekker, New York
- [396] Reich J.G., Zinke I. (1974): *Analysis of kinetic and binding measurements, IV Redundancy of model parameters*, Studia Biophysics, Vol. 43, 91-107
- [397] Rektorys K. (1982): *The Method of Discretization in Time and Partial Differential Equations*, Reidel, Dordrecht
- [398] Renardy M., Rogers R.C. (1993): *An Introduction to Partial Differential Equations*, Texts in Applied Mathematics, Vol. 13, Springer, Berlin
- [399] Rhee, K.I., Sohn H.Y. (1989): *The selective carbochlorination of iron from titaniferous magnetite ore in a fluidized bed*, Matallurgical Transactions B, Vol. 21B, 341-347
- [400] Richter O., Diekkruieger B., Noertersheuser P. (1996): *Environmental Fate Modelling of Pesticides*, VCH, Weinheim
- [401] Richter O., Noertersheuser, Pestemer W. (1992): *Non-linear parameter estimation in pesticide degradation*, The Science of the Total Environment, Vol. 123/124, 435-450
- [402] Richter O., Söndgerath D. (1990): *Parameter Estimation in Ecology*, VCH, Weinheim
- [403] Richter O., Spickermann U., Lenz F. (1991): *A new model for plant growth*, Gartenbauwissenschaft, Vol. 56, No. 3, 99-106
- [404] Ricker W. (1954): *Stock and recruitment*, Journal of Fishery Research Board Canada, Vol. 211, 559-663
- [405] Robertson H.H. (1966): *The solution of a set of reaction rate equations*, in: Numerical Analysis, J. Walsh ed., Academic Press, London, New York, 178-182
- [406] Roberson R.E., Schwertassek R. (1988): *Dynamics of Multibody Systems*, Springer, Berlin
- [407] Rominger K.L., Albert H.J. (1985): *Radioimmunological determination of Fenoterol. Part I: Theoretical fundamentals*, Arzneimittel-Forschung/Drug Research, Vol. 35, No. 1, 415-420
- [408] Roos Y.H. (1995): *Phase Transition in Foods*, Academic Press, San Diego
- [409] Rosenau P., Ströder A.C., Stirbet A.D., Strasser R.J. (1999): *Recent advances in modelling the photosynthesis*, Report, IWR, University of Heidelberg
- [410] Rosenbrock H.H. (1969): *An automatic method for finding the greatest and least value of a function*, Computer Journal, Vol. 3, 175-183
- [411] Ross G.J.S. (1990): *Nonlinear Estimation*, Springer, Berlin
- [412] Rudolph P.E., Herrendörfer G. (1995): *Optimal experimental design and accuracy of parameter estimation for nonlinear regression models used in long-term selection*, BNiomedical Journal, Vol. 37, 183-190

- [413] Runge C. (1895): *Über die numerische Auflösung totaler Differetialgleichungen*, Mathematische Annalen, Vol. 46, 167-178
- [414] Ryan T.P. (2007): *Modern Experimental Design*, Wiley Series in Probability and Statistics
- [415] Saad M.F., Anderson R.L., Laws A., Watanabe R.M., Kades W.W., Chen Y.-D.I. , Sands R.E., Pei D., Bergmann R.N. (1994): *A comparison between the minimal model and the glucose clamp in the assessment of insulin sensitivity across the spectrum of glucose tolerance*, Diabetes, Vol. 43, 1114-1121
- [416] Sakawa Y., Shindo Y. (1982): *Optimal control of container cranes*, Automatica, Vol. 18, 257-266
- [417] Sanz-Serna J.M., Calvo M.P. (1994): *Numerical Hamiltonian Processes*, Chapman and Hall, London
- [418] Saravacos G.D., Charm S.E. (1962): *A study of the mechanism of fruit and vegetable dehydration*, Food Technology, 78-81
- [419] Schenk J.L., Staudinger G. (1989): *Computer model of pyrolysis for large coal particles*, in: Proceedings of the International Conference of Coal Science, Tokyo
- [420] Schiesser W.E. (1991): *The Numerical Method of Lines*, Academic Press, New York, London
- [421] Schiesser W.E. (1994): *Computational Mathematics in Engineering and Applied Science*, CRC Press, Boca Raton
- [422] Schiesser W.E. (1994): *Method of lines solution of the Korteweg-de Vries equation*, Computers in Mathematics and Applications, Vol. 28, No. 10-12, 147-154
- [423] Schiesser W.E., Silebi C.A. (1997): *Computational Transport Phenomena*, Cambridge University Press
- [424] Schittkowski K. (1979): *Numerical solution of a time-optimal parabolic boundary-value control problem*, Journal of Optimization Theory and Applications, Vol. 27, 271-290
- [425] Schittkowski K. (1980): *Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer, Berlin
- [426] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction*, Mathematische Operationsforschung und Statistik, Ser. Optimization, Vol. 14, 197-216
- [427] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems*, Annals of Operations Research, Vol. 5, 485-500
- [428] Schittkowski K. (1987): *More Test Examples for Nonlinear Programming*, Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer, Berlin

- [429] Schittkowski K. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method*, in: Trends in Mathematical Optimization, K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, Boston, Basel, Berlin, 295-309
- [430] Schittkowski K. (1992): *Solving nonlinear programming problems with very many constraints*, Optimization, Vol. 25, 179-196
- [431] Schittkowski K. (1993): *DFDISC: A direct search Fortran subroutine for nonlinear programming*, User's Guide, Dept. of Mathematics, University of Bayreuth, Germany
- [432] Schittkowski K. (1994): *Parameter estimation in systems of nonlinear equations*, Numerische Mathematik, Vol. 68, 129-142
- [433] Schittkowski K. (1997): *Parameter estimation in one-dimensional time dependent partial differential equations*, Optimization Methods and Software, Vol. 7, No. 3-4, 165-210
- [434] Schittkowski K. (1998): *Parameter estimation in a mathematical model for substrate diffusion in a metabolically active cutaneous tissue*, Progress in Optimization II, 183 - 204, Proceedings of the Optimization Day, Perth, Australia, June 29-30
- [435] Schittkowski K. (1998): *Parameter estimation and model verification in systems of partial differential equations applied to transdermal drug delivery*, Report, Dept. of Mathematics, University of Bayreuth, Germany
- [436] Schittkowski K. (1999): *PDEFIT: A FORTRAN code for parameter estimation in partial differential equations*, Optimization Methods and Software, Vol. 10, 539-582
- [437] Schittkowski K. (2002): *EASY-FIT: A software system for data fitting in dynamic systems*, Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169
- [438] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems - A Practical Introduction with Applications and Software*, Kluwer Academic Publishers, Dordrecht, Boston, London
- [439] Schittkowski K. (2004): *PCOMP: A modeling language for nonlinear programs with automatic differentiation*, in: *Modeling Languages in Mathematical Optimization*, J. Kallrath ed., Kluwer, Norwell, MA, 349-367
- [440] Schittkowski K. (2006): *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 2.2*, Report, Department of Computer Science, University of Bayreuth
- [441] Schittkowski K. (2007): *Experimental design tools for ordinary and algebraic differential equations*, Industrial and Engineering Chemistry Research, Vol. 46, 9137-9147
- [442] Schittkowski K. (2007): *NLPQLB: A Fortran implementation of an SQP algorithm with active set strategy for solving optimization problems with a very large number of constraints - user's guide, version 2.0*, Report, Department of Computer Science, University of Bayreuth

- [443] Schittkowski K. (2007): *An active set strategy for solving optimization problems with up to 60,000,000 nonlinear constraints*, submitted for publication
- [444] Schittkowski K. (2007): *Parameter identification in one-dimensional partial differential algebraic equations*, GAMM-Mitteilungen, Vol. 30, No. 2, 352-375
- [445] Schittkowski K. (2007): *NLPMMX: A Fortran implementation of an SQP algorithm for min-max optimization - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth
- [446] Schittkowski K. (2007): *NLPLSQ: A Fortran implementation of an SQP-Gauss-Newton algorithm for least squares optimization - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth
- [447] Schittkowski K. (2008): *Parameter identification and model verification in systems of partial differential equations applied to transdermal drug delivery*, to appear: Mathematics and Computers in Simulation
- [448] Schittkowski K. (2008): *NLPINF: A Fortran implementation of an SQP algorithm for maximum-norm optimization problems - user's guide, version 2.0*, Report, Department of Computer Science, University of Bayreuth
- [449] Dai Y.-H., Schittkowski K. (2008): *A sequential quadratic programming algorithm with non-monotone line search*, Pacific Journal of Optimization, Vol. 4, 335-351
- [450] Schittkowski T., Brüggemann, Mewes B. (2002): *LII and Raman measurements in sooting methane and ethylene flames*, Report, LTTT, Dept. of Applied Natural Sciences, University of Bayreuth
- [451] Schneider R., Posten C., Munack A. (1992): *Application of linear balance equations in an online observation system for fermentation processes*, Proceedings of the IFAC Modelling and Control of Biotechnical Processes, Boulder, Colorado, 319-322
- [452] Schreiner T. (1995): *Mechanistische und kinetische Parameter der Arzneistoffauflösung aus festen Zubereitungen als Kriterien der galenischen Qualitätssicherung*, Dissertation, Dept. of Pharmaceutics, University of Saarbrücken
- [453] Schumacher E. (1997): *Chemische Reaktionskinetik*, Script, Dept. of Chemistry, University of Bern, Switzerland
- [454] Schwartz A.L. (1996): *Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems*, Ph.D. Thesis, Dept. of Electrical Engineering and Computer Sciences, University of California at Berkeley
- [455] Scott M.R., Watts H.A. (1976): *Solution methods for stiff differential equations*, in: Numerical Methods for Differential Systems, L. Lapidus, W.E. Schiesser eds., Academic Press, New York, London, 197-227
- [456] Seber G.A.F. (1977): *Linear Regression Analysis*, John Wiley, New York

- [457] Seber G.A.F. (1984): *Multivariate Observations*, John Wiley, New York
- [458] Seber G.A.F., Wild C.J. (1989): *Nonlinear Regression*, John Wiley, New York
- [459] Seelig F.F. (1981): *Unrestricted harmonic balance II. Application to stiff ODE's in enzyme catalysis*, Journal of Mathematical Biology, Vol. 12, 187-198
- [460] Seifert P. (1990): *A realization of the method of lines used for chemical problems*, Colloquia Mathematica Societatis Janos Bolyai, Numerical Methods, Vol. 59, 363-373
- [461] Sellers P.J., Dickinson R.E., Randall D.A., Betts A.K., Hall F.G., Berry J.A., Collatz G.J., Denning A.S., Mooney H.A., Nobre C.A., Sato N., Field C.B., Henderson-Sellers A. (1997): *Modeling the exchanges of energy, water, and carbon between continents and the atmosphere*, Science, Vol. 275, 502-509
- [462] Seredynski F. (1973): *Prediction of plate cooling during rolling mill operation*, Journal of the Iron and Steel Institute, Vol. 211, 197-203
- [463] Severo A.M., Diaz-Romanach M.L.B., Rodriguez L.M.P., Ginart J.B. (2004): *Numerical experience in the solution of several kind estimation problems in dynamical systems*, International Conference on Modeling and Optimization, MODOPT 2003, Tmuco, Chile, 16.-22.1.2004
- [464] Seydel R. (1988): *From Equilibrium to Chaos: Practical Bifurcation and Stability Analysis*, Elsevier, Amsterdam
- [465] Shacham M. (1985): *Comparing software for the solution of systems of nonlinear algebraic equations arising in chemical engineering*, Computers and Chemical Engineering, Vol. 9, 103-112
- [466] Shakhno, S. (2001): *Some numerical methods for nonlinear least squares problems*, in: Symbolic Algebraic Methods and Verification Methods, Alefeld, Götz eds., Springer, Wien, 235-249
- [467] Shampine L.F. (1980): *Evaluation of a test set for stiff ODE solvers*, ACM Transactions on Mathematical Software, Vol. 7, No. 4, 409-420
- [468] Shampine L.F. (1994): *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, New York, London
- [469] Shampine L.F., Watts H.A., Davenport S.M. (1976): *Solving nonstiff ordinary differential equations - The state of the art*, SIAM Reviews, Vol. 18, 376-411
- [470] Shampine L.F., Watts H.A. (1979): *The art of writing a Runge-Kutta code*, Applied Mathematics and Computations, Vol. 5, 93-121
- [471] Shampine L.F., Gordon M.K. (1975): *Computer Solution of Ordinary Differential Equations: The Initial-Value Problem*, Freeman, San Francisco

- [472] Sheng Q., Khalic A.Q.M. (1999): *A compound adaptive approach to degenerate nonlinear quenching problems*, Numerical Methods for Partial Differential Equations, Vol. 16, No. 1, 107-132
- [473] Shiriaev D., Griewank A., Utke J. (1997): *A user guide to ADOL-F: Automatic differentiation of Fortran codes*, Preprint, Institute of Scientific Computing, Technical University Dresden, Germany
- [474] Shu C.W. (1998): *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in: Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, A. Quarteroni ed., Lecture Notes in Mathematics, Vol. 1697, Springer, Berlin, 325-432
- [475] Shu C.W., Osher S. (1989): *Efficient implementation of essentially non-oscillatory shock-capturing schemes, II*, Journal on Computational Physics, Vol. 83, 32-78
- [476] Silver S. (1949): *Microwave Antenna Theory and Design*, McGraw Hill, New York
- [477] Simeon B. (1994): *Numerische Integration mechanischer Mehrkörpersysteme: Projizierende Deskriptorformen, Algorithmen und Rechenprogramme*, Fortschrittberichte VDI, Reihe 20, Nr. 130, VDI, Düsseldorf
- [478] Simeon B., Rentrop P. (1993): *An extended descriptor form for the simulation of constrained mechanical systems*, in: Advanced Multibody System Dynamics, W. Schiehlen ed., Kluwer Academic Publishers, Dordrecht, Boston, London, 469-474
- [479] Simeon B., Grupp F., Führer C., Rentrop P. (1994): *A nonlinear truck model and its treatment as a multibody system*, Journal of Computational and Applied Mathematics, Vol. 50, 523-532
- [480] Sincovec R.F., Madsen N.K. (1975): *Software for nonlinear partial differential equations*, ACM Transactions on Mathematical Software, Vol. 1, No. 3, 232-260
- [481] Slider H.C. (1976): *Practical Petroleum Reservoir Engineering Methods*, The Petroleum Publishing Company, Tulsa, Oklahoma
- [482] Smith G.D. (1985): *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Clarendon Press, Oxford Applied Mathematics and Computing Science Series
- [483] Smith M.G. (1966): *Laplace Transform Theory*, Van Nostrand
- [484] Smoller J. (1994): *Shock Waves and Reaction-Diffusion Equations*, Grundlehren der mathematischen Wissenschaften, Vol. 258, Springer, Berlin
- [485] Spellucci P. (1993): *Numerische Verfahren der nichtlinearen Optimierung*, Birkhäuser, Boston, Basel, Berlin
- [486] Spellucci P. (1998): *A SQP method for general nonlinear programs using only equality constrained subproblems*, Mathematical Programming, Vol. 82, 413-448

- [487] Spicer P.T., Pratsinis S.E. (1996): *Coagulation and fragmentation: universal steady-state particle-size distribution*, *AIChE Journal*, Vol. 42, No. 6, 1612-1620
- [488] Spiegel M.R. (1965): *Laplace Transforms*, Schaum's Outline Series, McGraw Hill, New York
- [489] Spoelstra J., van Wyk D.J. (1987): *A method of solution for a non-linear diffusion model and for computing the parameters in a model*, *Journal of Computational and Applied Mathematics*, Vol. 20, 379-385
- [490] Stehfest H. (1970): *Algorithm 368: Numerical inversion of Laplace transforms*, *Communications of the ACM*, Vol. 13, 47-49
- [491] Steinebach G., Rentrop P. (2000): *An adaptive method of lines approach for modelling flow and transport in rivers*, Preprint No. 00/09, IWRMM, University of Karlsruhe
- [492] Steinsträsser I. (1994): *The organized HaCaT cell culture sheet: A model approach to study epidermal peptide drug metabolism*, Dissertation, Pharmaceutical Institute, ETH Zürich
- [493] Stenger F., Gustafson S.-A., Keyes B., O'Reilly M., Parker K. (1999): *ODE-IVP-PACK via Sinc indefinite integration and Newton's method*, *Numerical Algorithms*, Vol. 20, 241-268
- [494] Stirbet A.D., Strasser R.J. (1996): *Numerical solution of the in vivo fluorescence in plants*, *Mathematical Computations and Simulations*, Vol. 42, 245-253
- [495] Stoer J. (1985): *Foundations of recursive quadratic programming methods for solving nonlinear programs*, in: *Computational Mathematical Programming*, K. Schittkowski ed., NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 15, Springer, Berlin
- [496] Stoer J., Bulirsch R. (1980): *Introduction to Numerical Analysis*, Springer, New York
- [497] Stortelder W.J.H. (1998): *Parameter estimation in nonlinear dynamical systems*, Dissertation, National Research Institute for Mathematics and Computer Science, University of Amsterdam
- [498] Strikwerda J.C. (1997): *Finite Difference Schemes and Partial Differential Equations*, Chapman and Hall, New York
- [499] Swameye I., Müller T.G., Timmer J., Sandra O., Klingmüller U. (2002): *Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by database modeling*, *Proceedings of the National Society of Sciences of the USA*, Vol. 100, 1028-1033
- [500] Sweby P.K. (1984): *High resolution schemes using flux limiters for hyperbolic conservation laws*, *SIAM Journal on Numerical Analysis*, Vol. 21, No. 5, 995-1011
- [501] Tanaka Y., Fukushima M., Ibaraki T. (1988): *A comparative study of several semi-infinite nonlinear programming algorithms*, *European Journal of Operations Research*, Vol. 36, 92-100
- [502] Teo K.L., Wong K.H. (1992): *Nonlinearly constrained optimal control of nonlinear dynamic systems*, *Journal of the Australian Mathematical Society, Ser. B*, Vol. 33, 507-530

- [503] Thomas J.W. (1995): *Numerical Partial Differential Equations*, Texts in Applied Mathematics, Vol. 22, Springer, Berlin
- [504] Thomaseth K., Cobelli C. (1999): *Generalized sensitivity functions in physiological system identification*, Annals of Biomedical Engineering, Vol. 27, 607-616
- [505] Thomopoulos S.C.A., Papadakis I.N.M. (1991): *A single shot method for optimal step computation in gradient algorithms*, Proceedings of the 1991 American Control Conference, Boston, MA, American Control Council, IEEE Service Center, Piscataway, NJ, 2419-2422
- [506] Timmer J., Rust H., Horbelt W., Voss H.U. (2000): *Parametric, nonparametric and parametric modelling of a chaotic circuit time series*, Physics Letters A, Vol. 274, 123-134
- [507] Timoshenko S., Goodier J.N. (1970): *Theory of Elasticity*, McGraw Hill, New York
- [508] Tjoa I., Biegler L. (1991): *Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems*, Industrial Engineering Chemistry Research, Vol. 30, 376-385
- [509] Tjoa T.B., L.T. Biegler L.T. (1992): *Reduced successive quadratic programming strategy for errors-in-variables estimation*, Computers and Chemical Engineering, Vol. 16, 523
- [510] Troeltzsch F. (1999): *Some remarks on second order sufficient optimality conditions for nonlinear elliptic and parabolic control problems*, in: Proceedings of the Workshop 'Stabilität und Sensitivität von Optimierungs- und Steuerungsproblemen', Burg (Spreewald), Germany, 21.-23.4.99
- [511] Tveito A., Winther R. (1998): *Introduction to Partial Differential Equations*, Springer, New York
- [512] Tzafrini A.R. (2000): *Mathematical modelling of diffusion-mediated release from bulk degrading matrices*, Journal of Controlled Release, Vol. 63, 69-79
- [513] Ulbrich S. (1995): *Stabile Randbedingungen und implizite entropiedissipative numerische Verfahren für Anfangs-Randwertprobleme mehrdimensionaler nichtlinearer Systeme von Erhaltungsgleichungen mit Entropie*, Dissertation, TU München, Institut für Angewandte Mathematik und Statistik
- [514] van den Bosch B.A.J. (1978): *Identification of parameters in distributed chemical reactors*, in: Distributed Parameter Systems, W.H. Ray, D.G. Lainiotis eds., Marcel Dekker, New York, Basel, 47-134
- [515] van den Bosch P.P.J., van der Klauw A.C. (1994): *Modeling, Identification and Simulation of Dynamical Systems*, CRC Press, Boca Raton, Ann Arbor, London, Tokyo
- [516] van Doesburg H., De Jong W.A. (1974): *Dynamic behavior of an adiabatic fixed-bed methanator*, in: Advances in Chemistry, Vol. 133, International Symposium on Reaction Engineering, Evanston, 489-503

- [517] van Duijn C.J. (1989): *Flow through porous media*, DFG-SPP-Report No. 135, Dept. of Mathematics, University of Augsburg
- [518] van Genuchten M.T. (1980): *A closed-form equation for predicting the hydraulic conductivity of unsaturated soils*, Soil Science Society of America Journal, Vol. 44, 892-898
- [519] van Genuchten M.T., Wierenga P.J. (1976): *Mass transfer studies in sorbing porous media. 1. Analytical solutions*, Soil Sciences Society of America Journal, Vol. 44, 892-898
- [520] van Kan J.J.I.M., Segal A. (1995): *Numerik partieller Differentialgleichungen für Ingenieure*, Teubner
- [521] Vande Wouwer A., Saucec Ph., Schiesser W.E. (2001): *Adaptive Methods of Lines*, Chapman and Hall/CRC, Boca Raton
- [522] Varah J.M. (1982): *A spline least squares method for numerical parameter estimation in differential equations*, SIAM Journal on Scientific Statistical Computing, Vol. 3, 28-46
- [523] Varma A., Morbidelli M., Wu H. (1999): *Parametric Sensitivity in Chemical Systems*, Cambridge University Press
- [524] Vasile M., Jehn R. (1999): *Low thrust orbital transfer of a LISA spacecraft with constraints on the solar aspect angle*, MAS Working Paper 424, ESOC, Darmstadt
- [525] Vassiliadis V.S., Sargent R.W.H., Pantelides C.C. (1994): *Solution of a class of multistage dynamic optimization problems, 2. Problems with path constraints*, Industrial Engineering and Chemical Research, Vol. 33, No. 9, 2123-2133
- [526] Versyck K.J., Claes J., Van Impe J. (1997): *Practical identification of unstructured growth kinetics by application of optimal experimental design*, Biotechnical Progress, Vol. 13, 524-531
- [527] Verwer J.G., Blom J.G., Fuzeland R.M., Zegeling P.A. (1989): *A moving grid method for one-dimensional PDEs based on the method of lines*, in: Adaptive Methods for Partial Differential Equations, J.E. Flaherty, P.J. Paslow, M.S. Shephard, J.D. Vasilakis eds., SIAM, Philadelphia, Pa., 160-175
- [528] Verwer J.G., Blom J.G., Sanz-Serna J.M. (1989): *An adaptive moving grid method for one-dimensional systems of partial differential equations*, Journal of Computational Physics, Vol. 82, 454-486
- [529] Vlassenbeck J., van Dooren R. (1983): *Estimation of the mechanical parameters of the human respiratory system*, Mathematical Biosciences, Vol. 69, 31-55
- [530] von Stryck O. (1995): *Numerische Lösung optimaler Steuerungsprobleme*, Fortschrittsberichte VDI, Reihe 8, Nr. 441, VDI, Düsseldorf
- [531] Vossen G., Rehbock V., Siburian A. (2005): *Numerical solution methods for singular control with multiple state dependent forms*, submitted for publication

- [532] Vrar, C.K. (2000): *The study of kinetics and mechanism of chlorination of copper(I)sulphide by chlorine in the presence of oxygen*, Canadian Metallurgical Quarterly, Vol. 39, 163-174
- [533] Vreugdenhil C.B., Koren B. eds. (1993): *Numerical Methods for Advection-Diffusion Problems*, Vieweg, Braunschweig
- [534] Walas S.M. (1991): *Modeling with Differential Equations in Chemical Engineering*, Butterworth-Heinemann, Boston
- [535] Waldron R.A. (1969): *Theory of Guided Electromagnetic Waves*, Van Nostand Reinhold Company, London
- [536] Walsteijn F.H. (1993): *Essentially non-oscillatory (ENO) schemes*, in: Numerical Methods for Advection-Diffusion Problems, C.B. Vreugdenhil, B. Koren eds., Notes on Fluid Mechanics, Vol. 45, Vieweg, Braunschweig
- [537] Walter E. (1982): *Identifiability of State Space Models*, Lecture Notes in Biomathematics, Vol. 46, Springer, Berlin
- [538] Walter E., Pronzato L. (1997): *Identification of Parametric Models*, Springer, Paris, Milan, Barcelone
- [539] Walter S., Lorimer G.H., Schmid F.X. (1996): *A thermodynamic coupling mechanism for GroEl-mediated unfolding*, Biochemistry, Vol. 93, 9425-9430
- [540] Wang H., Al-Lawatia M., Sharpley R.C. (1999): *A characteristic domain decomposition and space-time local refinement method for first-order linear hyperbolic equations with interface*, Numerical Methods for Partial Differential Equations, Vol. 15, No. 1, 1-28
- [541] Wang Z., Richards B.E. (1991): *High resolution schemes for steady flow computation*, Journal of Computational Physics, Vol. 97, 53-72
- [542] Wansbrough R.W. (1985): *Modeling chemical reactors*, Chemical Engineering, Vol. 5, 95-102
- [543] Watts D.G. (1981): *An introduction to nonlinear least squares*, in: Kinetic Data Analysis: Design and Analysis of Enzyme and Pharmacokinetic Experiments, L. Endrenyi ed., Plenum Press, New York, 1-24
- [544] Weinreb A., Bryson A.E.Jr. (1985): *Optimal control of systems with hard control bounds*, IEEE Transactions on Automatic Control, Vol. AC-30, 1135-1138
- [545] Weizhong D., Nassar R. (1999): *A finite difference scheme for solving the heat transport equation at the microscale*, Numerical Methods for Partial Differential Equations, Vol. 15, No. 6, 697-708
- [546] Wen C.S., Yen T.F. (1977): *Optimization of oil shale pyrolysis*, Chemical Engineering Sciences, Vol. 32, 346-349

- [547] Williams M.L., Landel R.F., Ferry J.D. (1955): *The temperature dependence of relaxation mechanisms in amorphous polymers and other glass-forming liquids*, Journal of the American Chemical Society, Vol. 77, 3701-3706
- [548] Williams J., Kalogiratou Z. (1993): *Least squares and Chebyshev fitting for parameter estimation in ODE's*, Advances in Computational Mathematics, Vol. 1, 357-366
- [549] Willoughby, R.A. (1974): *Stiff Differential Systems*, Plenum Press, New York
- [550] Widder D.V. (1941): *The Laplace Transform*, Princeton University Press
- [551] Winer B.J., Brown D.R., Michels K.M. (1971): *Statistical Principles In Experimental Design*, McGraw-Hill
- [552] Wolf M. (1994): *Mathematisch-physikalische Berechnungs- und Simulationsmodelle zur Beschreibung und Entwicklung transdermaler Systeme*, Habilitationsschrift, Mathematisch-Naturwissenschaftliche Fakultät, Universität Bonn
- [553] Wolf H., Sauerer B., Fasold D., Schlesinger V. (1994): *Computer aided optimization of circular corrugated horns*, Proceedings of the Progress in Electromagnetics Research Symposium, Noordwijk, The Netherlands
- [554] Wolmott P., Dewynne J.N., Howison S.D. (1993): *Option Pricing: Mathematical Models and Computation*, Oxford Financial Press
- [555] Wouwer A.V. (1994): *Simulation, parameter and state estimation techniques for distributed parameter systems with real-time application to a multizone furnace*, Dissertation, Faculte Polytechnique de Mons, Belgium
- [556] Yang H.Q., Przekwas A.J. (1992): *A comparative study of advanced shock-capturing schemes applied to Burgers' equation*, Journal of Computational Physics, Vol. 102, 139-159
- [557] Yee H.C. (1985): *Construction of a class of symmetric TVD schemes*, Lectures in Applied Mathematics, Vol. 22, 381-395, Springer, Berlin
- [558] Zachmanoglou E.C., Thoe D.W. (1986): *Introduction to Partial Differential Equations with Applications*, Dover
- [559] Zeeman E.C. (1972): *Differential equations for the heartbeat and nerve impulse*, in: Towards a Theoretical Biology, C.H. Waddington ed., Edinburgh University Press, Vol. 4, 8-67
- [560] Zegeling P.A., Verwer J.G., van Eijkeren J.C.H. (1992): *Application of a moving grid method to a class of 1d brine transport problems in porous media*, International Journal for Numerical Methods in Fluids, Vol. 15, 175-191
- [561] Zhengfeng L., Osborne M.R., Prvan T. (2002): *Parameter estimation of ordinary differential equations*, to appear: IMA Journal of Numerical Analysis

- [562] Zscheschang, T. Dresig, H. (1998): *Zur Zeit-Frequenz-Analyse von Schwingungen in Antrieben von Verarbeitungsmaschinen*, Fortschrittsberichte VDI, Nr. 1416, VDI, Düsseldorf, 489-506