EASY-FIT: A Software System for Data Fitting in Dynamical Systems

Prof. K. Schittkowski
Department of Mathematics
University of Bayreuth
95440 Bayreuth, Germany

July 6, 2000

Abstract

EASY-FIT is an interactive software system to identify parameters in explicit model functions, steady-state systems, 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.

The software system is implemented in form of a Microsoft Access database running under MS-Windows 95/98/NT. 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 Fortran-similar modeling language, that allows in addition automatic differentiation of nonlinear functions, or by user-provided Fortran subroutines.

Keywords: parameter estimation, data fitting, dynamical systems, ODE, DAE, PDE

1 Introduction

Data fitting plays an important role in many natural science, engineering and other disciplines. The key idea is to estimate unknown parameters in a mathematical model that describes a real life situation, by minimizing the distance of some known experimental data from the theoretically predicted model function 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. To sum up, parameter estimation or data fitting, respectively, is extremely important in all practical situations, where a mathematical model and corresponding experimental data are available to describe the behaviour of a dynamical system.

The purpose of the paper is to introduce an interactive software system called EASY-FIT that performs parameter estimation by a least squares fit. The mathematical model has to belong to one of the following categories:

- explicit model functions
- dynamical systems of nonlinear equations (steady-state)
- Laplace transformations of differential equations
- ordinary differential equations with initial values
- differential algebraic equations
- one-dimensional, time-dependent partial differential equations
- one-dimensional partial differential algebraic equations

Model functions may depend also on an additional independent variable that could represent for example a concentration value of an experiment. 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*. This notation reflects the probably most typical applications. On the other hand, these terms may have any other meaning within a model depending on the underlying application.

Nonlinear model functions are evaluated symbolically when implemented in the available modeling language. Compilation and link of Fortran subroutines is not required whenever model functions are defined or altered in this way. A particular advantage of the approach is automatic differentiation of model functions to avoid numerical truncation errors. The corresponding program is called PCOMP (Dobmann, Liepelt, Schittkowski, 1995), and is part of the executable codes.

The development of EASY-FIT goes back to 1980, when the author performed a major comparative performance evaluation of nonlinear programming codes (Schittkowski, 1980). Convinced on the success of sequential quadratic programming algorithms, a Fortran code called NLPQL was developed a bit later (Schittkowski, 1985/86). Main applications of NLPQL are in structural mechanical engineering. Three years later the same code was the basis for an extension, to solve also least squares, L_1-, or min-max problems as efficiently as special purpose programs (Schittkowski, 1988). Motivated by some cooperative research projects with chemical and pharmaceutical firms, a first user interface of EASY-FIT was developed in 1990, still under DOS. This very first attempt was implemented a few years later in form of a MS-Access database with graphical user interface running under Windows. By taking the feedback of users into account, the GUI was steadily improved and extended until the present version. Parallel to the development of the user interface, also the numerical routines were improved and extended constantly, for example by introducing additional algebraic partial differential equations or hyperbolic transport equations, for which special discretization procedures were needed.

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

A brief review of available numerical algorithms is presented in Section 3. Only some basic features of the underlying ideas are presented. More details are found in the references. The codes allow the numerical identification of parameters in any of the seven situations under investigation. The executable files are called MODFIT.EXE and PDEFIT.EXE.

EASY-FIT is implemented in form a database for accessing model information, experimental data, and results, whereas all numerical routines are written in Fortran. A context sensitive help option is included containing additional technical and organizational information e.g. about the input of data and optimization tolerances. The organization of the software system and the menu-driven graphical user interface are outlined in Section 4.

EASY-FIT in permanent use to solve *real life* problem, i.e. problems with some realistic practical background. Application areas are pharmacy, biochemistry, chemical engineering, physics, and especially mechanical engineering, among others. A case study is presented in Section 5, where a model for cooling a hot strip mill is to be identified. A few further practical applications are listed in Section 6.

2 Parameter Estimation Models

EASY-FIT is an interactive software system to identify parameters in dynamical systems, i.e. parameters that are *hidden* in additional time-dependent state equations that must be solved implicitly. The basic mathematical structure is a least squares formulation of our data fitting problem, i.e. minimization of a sum of squares of nonlinear functions of the form

$$\begin{array}{ll} \min \sum_{r=1}^{l} f_r(p)^2 & \\ g_j(p) = 0\ , & j = 1, ..., m_e\ \ , \\ p \in I\!R^n: \ \ g_j(p) \geq 0\ , & j = m_e+1, ..., m\ \ , \\ p_l \leq p \leq p_u & . \end{array} \tag{1}$$

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 treated independently from the remaining constraints.

Our least squares parameter estimation algorithms proceed from the above formulation, although in the one or other case different approaches are available to define the objective functions. The assumption, that all problem functions must be smooth, is essential. The efficient numerical algorithm under consideration are based more or less on the Gauss-Newton method, that requires first derivatives. Alternatively the L_2-norm may be changed to the L_1- or the L_∞-norm.

To get a data fitting problem, we suppose that one vector-valued model function $h(p, t, c)$ is available, the so-called *fitting criterion function*, depending on the parameter vector p to be identified, furthermore on an additional variable t called time, and optionally on another one c called *concentration*. Both are also called the independent model variables, and h the dependent one.

Now we proceed from n_r sets of experimental measurements in the form

$$(t_i, c_j, y_{ij}^k)\ ,\ i = 1, \ldots, n_t,\ j = 1, \ldots, n_c,\ k = 1, \ldots, n_r\ , \tag{2}$$

where n_t time values, n_c concentration values and $n_t n_c n_r$ corresponding experimental measurement values are defined. Together with the vector-valued model function

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

we get the above least squares formulation by defining

$$f_r(p) = w_{ij}^k (h_k(p,t_i,c_j) - y_{ij}^k) \quad , \tag{3}$$

where r runs from 1 to $l = n_t n_c n_r$ in any order. Moreover we assume that there are suitable weighting factors $w_{ij}^k > 0$ given by the user, that are to reflect the individual influence of a measurement for the whole experiment.

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

If we assume that $h(p,t,c)$ does not depend on the solution of additional state equations, we call it an explicit model function. Otherwise $h(p,t,c)$ may depend in addition on the solution vector of an auxiliary problem, e.g. an ordinary differential equation, that is implicitly defined. Models of this kind are considered in the subsequent sections.

2.1 Steady-State Systems of Equations

In this case the model function $h(p,t,c)$ depends in addition on the solution vector of a system of nonlinear equations, i.e.

$$h(p,t,c) = \overline{h}(p, z(p,t,c), t, c) \quad , \tag{4}$$

where $z(p,t,c) \in I\!R^s$ is implicitly defined by the solution z of the nonlinear system

$$\begin{aligned} G_1(p,z,t,c) &= 0 \ , \\ &\cdots \\ G_s(p,z,t,c) &= 0 \ . \end{aligned} \tag{5}$$

These systems of equations arise, for example, if the left-hand side of a differential equation $\dot{z_j} = G_j(p,z,t,c)$ for $j = 1, \ldots, s$ is set to zero by assuming, that a steady-state is reached. Usually the time variable t has another physical meaning in this case, e.g., a temperature or a concentration, and c may be any other independent model variable.

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

$$\left(\frac{\partial G_i(p,z,t,c)}{\partial z_j} \right)_{i,j=1,\ldots,s}$$

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 $z(p,t,c)$ is differentiable with respect to all p in the feasible domain. The

Jacobian matrix of the solution z of the state equation w.r.t. the parameters to be estimated, is easily obtained from (5) by solving the linear system

$$\nabla_p G(p, z(p,t,c)) + \nabla_z G(p, z(p,t,c)) V = 0 \quad , \tag{6}$$

where V is a $s \times n$-matrix.

2.2 Laplace Models

In many practical applications, the model is available in form of a Laplace formulation to simplify the underlying analysis. The numerical algorithms described in this paper, are able to proceed directly from the Laplace transform and to compute its inverse internally by a quadrature formula.

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

If the model function is given in form of a Laplace transform, say $H(p, s, c)$, the back-transformation is performed numerically by a quadrature method (Stehfest, 1970), and $h(p, t, c)$ is the numerical inverse Laplace transform of $H(p, s, c)$. We have to choose a formula that allows the exact computation of the gradients of the fitting functions w.r.t. the parameters to be estimated, if derivatives of the Laplace transform $\nabla_p H(p, s, c)$ are known.

2.3 Systems of Ordinary Differential Equations

The fitting criterion or the dependent model variable depends now the solution $y(p, t, c)$ of a system of ordinary differential equations, i.e.

$$h(p,t,c) = \overline{h}(p, y(p,t,c), t, c) \quad , \tag{7}$$

where $y(p, t, c)$ denotes the solution of a system of s ordinary differential equations

$$\begin{aligned} \dot{y}_1 = &\quad F_1(p,y,t,c) \quad , \; y_1(0) = y_1^0(p,c) \quad , \\ &\quad \cdots \\ \dot{y}_s = &\quad F_s(p,y,t,c) \quad , \; y_s(0) = y_s^0(p,c) \quad . \end{aligned} \tag{8}$$

Without loss of generality we assume that the initial time is zero. The initial values $y_1^0(p, c)$, ..., $y_s^0(p, c)$ 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.

There are many practical situations, where a model changes during integration over the time, and where initial values are to be adopted. A typical example is a chemical reactor model with non-continuous, time-dependent input functions. Thus, break or switching points τ_i with $0 < \tau_1 < \ldots < \tau_{n_b}$ are allowed, where the initial values for each sub-interval are given by any functions $y_j^i(p, c, y)$ 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 break point. It is possible that break points become variables to be adapted during the optimization process, if they are not known in advance.

Constraints of the form (1) are permitted, where the restriction functions may depend on the solution of the dynamical system at predetermined time and concentration values, i.e.

$$g_j(p) = \overline{g}_j(p, y(p, t_j, c_j), t_j, c_j) \tag{9}$$

for $j = 1, ..., m$. The m predetermined time and, if available, concentration values must coincide with some of the given measurement values. If not, the given data are rounded to the nearest experimental value. 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_j and c_j, respectively.

2.4 Systems of Differential Algebraic Equations

Parameter estimation problems based on differential algebraic equations, are very similar to those based on ordinary differential equations. The only difference is that we allow additional algebraic equations together with additional state variables.

Thus we have to replace (8) by the extended system

$$\begin{array}{lll} \dot{y}_1 = & F_1(p, y, z, t, c) & , \; y_1(0) = y_1^0(p, c) \;\; , \\ & \cdots & \\ \dot{y}_{s_1} = & F_{s_1}(p, y, z, t, c) & , \; y_{s_1}(0) = y_{s_1}^0(p, c) \;\; , \\ 0 = & G_1(p, y, z, t, c) & , \; z_1(0) = z_1^0(p, c) \;\; , \\ & \cdots & \\ 0 = & G_{s_2}(p, y, z, t, c) & , \; z_{s_2}(0) = z_{s_2}^0(p, c) \;\; . \end{array} \tag{10}$$

Now $y(p, t, c)$ and $z(p, t, c)$ are solution vectors of a joint system of $s_1 + s_2$ 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 w.r.t. z, i.e. if the matrix

$$\nabla_z G(p, y, z, t, c) \tag{11}$$

possesses full rank. In all other cases we get DAE's with a higher index, see (Hairer, Wanner, 1991) for a suitable definition and more details.

Similar to systems of ordinary differential equations, it is possible to define switching or break points, and to modify initial values at these points, that may depend on the parameters to be estimated, the concentration variable and the solution w.r.t. the previous interval.

For simplicity we consider now only problems of index 1, although the numerical algorithms are capable to treat also higher index models. Moreover problems with higher index can be transformed to problems of index 1 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

$$G_1(p, y^0(p,c), z^0(p,c), t, c) = 0 \ , \ \ldots \ , \ G_{s_2}(p, y^0(p,c), z^0(p,c), t, c) = 0 \ . \tag{12}$$

The initial values $y_1^0(p,c), \ldots, y_{s_1}^0(p,c)$ and $z_1^0(p,c), \ldots, z_{s_2}^0(p,c)$ are functions depending on the parameters to be estimated, and the concentration variable.

Parameter estimation problems based on DAE models, can be solved by the program MODFIT. The code checks, whether the consistency condition is satisfied or not when starting an integration cycle. In the latter case, consistent initial values are computed by solving the above nonlinear system of equations w.r.t. z after inserting initial values for the differential variables. Also if switching points exist, consistent initial values are evaluated before restarting the integration procedure. As before, additional dynamical nonlinear equality and inequality constraints can be taken into account.

2.5 Systems of One-Dimensional Time-Dependent Partial Differential Equations

Partial differential equation are extensions of ordinary differential equations, when an additional space or spatial variable x is introduced together with corresponding first and optionally also some higher order partial derivatives of the state variables. Again we assume without loss of generality, that the initial time is zero. This assumption facilitates the description of the mathematical model and is easily satisfied in practice by a suitable linear transformation of the time variable.

The model we want to investigate now, is defined by a system of n_p one-dimensional partial differential equations in one or more spatial intervals (Schittkowski, 1997). These intervals that could describe e.g. material areas with different diffusion coefficients, are given by the outer boundary values x_L and x_R that define the total integration interval w.r.t. the space variable x, and optionally some additional internal transition points $x_1^a, \ldots, x_{n_a-1}^a$. Thus we get a sequence of $n_a + 1$ boundary and transition points

$$x_0^a := x_L < x_1^a < \ldots < x_{n_a-1}^a < x_{n_a}^a := x_R \ . \tag{13}$$

For each integration interval, we define a one-dimensional, time-dependent partial differential equation of the form

$$u_t^i = F^i(x, t, f^i(x, t, v, u^i, u_x^i, p), f_x^i(x, t, v, u^i, u_x^i, p), v, u^i, u_x^i, u_{xx}^i, p) \ , \tag{14}$$

where $x \in I\!R$ is the spatial variable with $x_{i-1}^a \leq x \leq x_i^a$ for $i = 1, \ldots, n_a$, $t \in I\!R$ the time variable with $0 < t \leq T$, $v \in I\!R^{n_o}$ the solution vector of the coupled system of ordinary differential equations, $u^i \in I\!R^{n_p}$ the system variable we want to compute, and $p \in I\!R^n$ the parameter vector to be identified by the data fitting algorithm. Note that we omit the second independent model variable c for simplicity.

Optionally the right-hand side depends also on a so-called flux function $f^i(x, t, v, u^i, u_x^i, p)$, which is introduced either to facilitate the declaration of the function side or for being able to apply special discretization formulae in case of hyperbolic or related equations, when usual approximation schemes break down, e.g. when shocks propagate through the integration interval. In this case, the underlying system of equations is of the form

$$u_t^i = f_x^i(x,t,u^i,p) \ , \quad i = 1,\dots n_a \ . \tag{15}$$

For both end points x_L and x_R, we allow Dirichlet, Neumann, or mixed boundary conditions. Transition conditions between different areas may be defined in a very similar way, and depend on the solution values or their spatial derivatives of the left or right sub-interval at the transition point, respectively. Note that boundary information is also contained in coupled ordinary differential equations.

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

$$u^i(x,0,p) = u_0^i(x,p) \ , \quad i = 1,\dots,n_a \tag{16}$$

and are defined for all $x \in \left[x_{i-1}^a, x_i^a\right]$, $i = 1,\dots,n_a$.

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

$$\dot{v}_j = G_j(t,v,u^{i_j}(x_j,t,p), u_x^{i_j}(x_j,t,p), u_{xx}^{i_j}(x_j,t,p), p) \tag{17}$$

for $j = 1,\dots,n_o$ with initial values

$$v(0,p) = v_0(p) \ , \tag{18}$$

that may depend again on the parameters to be estimated. The system has n_o components, i.e. $v = (v_1,\dots,v_{n_o})^T$. 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 belongs to the j-th area, i.e. $x_j \in [x_{i_j-1}^a, x_{i_j}^a)$ or $x_j \in [x_{n_a-1}^a, x_{n_a}^a]$, respectively, $j = 1,\dots,n_o$, and is called coupling point.

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

Similar to systems of ordinary differential equations, it is possible to define break or switching points and modified initial values at these points, that may depend on the parameters to be estimated, the spatial variable and the solution w.r.t. the previous interval.

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

$$h_k(p,t) = \overline{h}_k(p,t,v(t,p),u^{i_k}(x_k,t,p),u_x^{i_k}(x_k,t,p),u_{xx}^{i_k}(x_k,t,p)) \tag{19}$$

and insert $\overline{h}_k$ instead of h_k into the data fitting function. Again the fitting criteria may depend on solution values at a given spatial variable value w.r.t. to an integration interval defined by the index i_k. The spatial variable x_k belongs to the i_k-th integration area, i.e. $x_k \in \left[x_{i_k-1}^a, x_{i_k}^a\right)$ or $x_k \in \left[x_{n_a-1}^a, x_{n_a}^a\right]$, respectively, $k = 1,\dots,n_r$, where n_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 w.r.t. the space variable at the fitting point. Fitting points are rounded to their nearest line when discretizing the system.

As for ordinary differential equations, dynamical constraints are allowed, where the restriction functions may depend on the solution of the partial differential equation and its first and second spatial derivatives at predetermined time and spatial values, and the solution of the coupled ordinary differential equation at predetermined time values, i.e.

$$g_j(p) = \overline{g}_j(x_j, t_j, p, v(t_j, p), u^{j_k}(x_j, t_j, p), u_x^{j_k}(x_j, t_j, p), u_{xx}^{j_k}(x_j, t_j, p)) \tag{20}$$

for $j = 1, ..., m$. Here u^{j_k} denotes the corresponding integration area that contains the spatial parameter x_j. The m predetermined time values must coincide with some of the given measurement values. If not, the given data are rounded to the nearest experimental data. The corresponding spatial values must coincide with a line obtained by the equidistant discretization of the underlying integration interval. 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 space coordinates t_j and x_j, respectively.

In order to achieve smooth fitting criteria and constraints, we assume that all model functions depend are continuously differentiable with respect to 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 available (Schittkowski, 1997), the demo version of EASY-FIT about 200 further PDE models.

2.6 Systems of One-Dimensional Partial Differential Algebraic Equations

PDAE's have 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

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

transformed into a second order equation by introducing additional variables

$$\begin{aligned} u_t &= f(x, t, u, w_{xx}, p) \ , \\ 0 &= w - u_{xx} \ , \end{aligned}$$

or distributed systems of the form

$$\begin{aligned} u_t^1 &= f_1(x, t, u^1, u^2, p) \ , \\ u_x^2 &= f_2(x, t, u^1, u^2, p) \ , \end{aligned}$$

with initial values $u^1(x, 0, p) = u_1(x, p)$, $u^2(0, t, p) = u_2(t, p)$, transformed into the PDAE

$$\begin{aligned} u_t^1 &= f_1(x, t, u^1, u^2, p) \ , \\ 0 &- u_x^2 - f_2(x, t, u^1, u^2, p) \end{aligned}$$

or

$$\begin{aligned} u_x^2 &= f_2(x, t, u^1, u^2, p) \ , \\ 0 &- u_x^1 - f_1(x, t, u^1, u^2, p) \ , \end{aligned}$$

respectively.

Thus we proceed again from a sequence of $n_a + 1$ boundary and transition points

$$x_0^a := x_L < x_1^a < \ldots < x_{n_a-1}^a < x_{n_a}^a := x_R \tag{21}$$

and consider the PDAE-system

$$\begin{aligned} u_t^i &= F_1^i(x,t,f^i(x,t,v,u^i,u_x^i,p),f_x^i(x,t,v,u^i,u_x^i,p),v,u^i,u_x^i,u_{xx}^i,p)\ , \\ 0 &= F_2^i(x,t,f^i(x,t,v,u^i,u_x^i,p),f_x^i(x,t,v,u^i,u_x^i,p),v,u^i,u_x^i,u_{xx}^i,p)\ , \end{aligned} \tag{22}$$

$i = 1, \ldots n_a$, where $x \in I\!R$ is the spatial variable with $x_{i-1}^a \le x \le x_i^a$ for $i = 1, \ldots, n_a$, $t \in I\!R$ the time variable with $0 < t \le T$, $v \in I\!R^{n_o}$ the solution vector of the coupled system of ordinary differential equations, $u^i \in I\!R^{n_p}$ the system variable we want to compute, and $p \in I\!R^n$ the parameter vector to be identified by the data fitting algorithm.

But now, the state variables u^i are divided into so-called differential variables and algebraic variables, i.e. $u^i = (u_1^i, u_2^i)^T$, where the number of algebraic variables is identical to the number of algebraic equations summarized in the vector F_2. Also we allow flux functions, switching points, constraints for parameters and state functions, and coupled ordinary differential equations. All these extensions are treated in the same way as for partial differential equations without algebraic equations.

However, we must handle initial and boundary conditions with more care. We have to guarantee, that at least the boundary and transition conditions satisfy the algebraic equations

$$0 = F_2^i(x_i^a,t,f^i(x_i^a,t,v,u^i,u_x^i,p),f_x^i(x_i^a,t,v,u^i,u_x^i,p),v,u^i,u_x^i,u_{xx}^i,p) \tag{23}$$

for $i = 1, \ldots n_a$. If initial conditions for discretized algebraic equations are violated, then the corresponding system of nonlinear equations is solved internally proceeding from initial values given. In other words, consistent initial values are computed automatically, where the given data serve as starting parameters for the nonlinear programming algorithm applied.

Thus, we allow only index-1-systems unless it is guaranteed, that consistent initial values for the discretized DAE are available.

3 Numerical Algorithms

EASY-FIT is the graphical user interface for the parameter estimation programs MODFIT and PDEFIT, that are also executable outside of EASY-FIT. 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 implemented within a Fortran subroutine. In this section we describe very briefly the underlying numerical algorithms that are implemented.

3.1 Data Fitting Algorithms

The parameter estimation programs contain interfaces for four different least squares algorithms, where only one code is also capable to solve L_1- and L_∞-problems, i.e. problems,

where the sum of absolute residual or the maximum of absolute residual values is to be minimized.

DFNLP: 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 (Schittkowski, 1988). 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 by introducing additional variables and constraints. In all three situations, the resulting optimization problem is solved by a standard sequential quadratic programming code called NLPQL (Schittkowski, 1985).

DN2GB: The subroutine is a frequently used unconstrained least squares algorithm (Dennis, Gay, Welsch, 1981). The mathematical method is also based on a combined Gauss-Newton and quasi-Newton approach.

DSLMDF: First successive line searches are performed along the unit vectors by comparing function values only. The one-dimensional minimization is based on a quadratic interpolation. After a search cycle, the Gauss-Newton-type method DFNLP is executed with a given number of iterations. If a solution is not obtained with sufficient accuracy, the search is repeated (Nickel, 1995).

NELDER: This is a very simple search method (Nelder, Mead, 1965). Successively edges of a simplex are exchanged by comparing function values only. The edge with highest function value is replaced by another one generated by a move through the center of the actual simplex to the other side.

Note that only DFNLP is able to take linear or nonlinear constraints into account. However, all other algorithms satisfy upper and lower bounds of the parameters to be estimated. The algorithms are also capable to solve problems with large residuals. The choice of algorithm NELDER is only useful when the other algorithms fail because of non-differentiable model functions, a very bad starting point or large round-off errors in the function evaluation.

3.2 Solving Systems of Dynamical Equations

The program MODFIT is executed to solve parameter estimation problems based with steady-state equations. To solve systems of nonlinear equations, they are treated as constraints of a general nonlinear programming problem and solved by the Fortran code NLPQL (Schittkowski, 1985). 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. Thus, the algorithm behaves like Newton's method for solving a system of equations.

Also initial values required for starting 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 value. Moreover, the gradients of the model function $\overline{h}(p, z(p,t,c), t, c)$ 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.

3.3 Numerical Laplace-Back-Transformation

MODFIT allows to solve parameter estimation problems, where the model functions are defined in the Laplace space. In this case, constraints are not allowed.

If an analytical back-transformation is not available, we have to apply a numerical quadrature formula (Bellman, 1966). In our case we use a simple formula (Stehfest, 1970) with the particular advantage, that derivatives w.r.t. the parameters to be estimated, are easily obtained from the derivatives of the Laplace function.

3.4 Numerical Algorithms for Ordinary Differential Equations

Differential equations must be formulated explicitly, and are solved by a couple of integration routines. For implicit methods, derivatives of the right-hand side of the differential equation are evaluated analytically using either user-provided derivatives or internal automatic differentiation. Within MODFIT it is possible to select among seven different solvers:

DOPRI5	:	Explicit Runge-Kutta method of order 4/5 (Dormand, Prince, 1981) as implemented by Hairer, Nørsett, and Wanner (Hairer, Nørsett, Wanner, 1993). Step-length is adapted internally.
DOP853	:	Explicit Runge-Kutta method of order 8 based on Dormand and Prince formula (Dormand, Prince, 1981), see (Hairer, Nørsett, Wanner, 1993). Local error estimation and step size control is based on embedded formulae of order 5 and 3.
ODEX	:	Extrapolation method based on GBS algorithm with variable order and step-size (Hairer, Nørsett, Wanner, 1993).
RADAU5	:	Implicit Radau-type Runge-Kutta method of order 5 for stiff equations (Hairer, Wanner, 1991).
SDIRK4	:	Diagonally implicit Runge-Kutta method with 5 stages (Hairer, Wanner, 1991).
SEULEX	:	Extrapolation algorithm based on linearly implicit Euler method (Hairer, Wanner, 1991).
IND-DIR	:	Runge-Kutta-Fehlberg method of order 4 to 5 (Shampine, Watts, 1979) with additional sensitivity analysis implemented (Benecke, 1993).

Note that the first six 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. In these cases gradients w.r.t. parameters to be estimated, are obtained by external numerical differentiation. Codes RADAU5, SDIRK4 and SEULEX are able to solve also stiff equations.

The last algorithm IND-DIR is capable to evaluate derivatives of the solution of the ODE internally w.r.t. the parameters to be estimated, where derivatives of the Runge-Kutta scheme are used.

3.5 Numerical Algorithms for Differential Algebraic Equations

Algebraic differential equations are solved by the implicit solvers RADAU5, SDIRK4, or SEULEX, see above. DAE's with an index greater than 1 can be solved only by RADAU5. 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 NLPQL (Schittkowski, 1985). The initial values given for the algebraic equations are used as starting values for NLPQL.

3.6 Numerical Algorithms for Partial Differential Equations

The underlying idea is, to transform the partial differential into a system of ordinary differential equations by discretizing the model functions w.r.t. the spatial variable x. This approach is known as the method of lines (Schiesser, 1991).

For the i-th integration interval of the spatial variable, we denote the number of discretization points by n_i, $i = 1, \ldots, n_a$. 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 $u(x, t, p)$ w.r.t. the spatial variable at a given point x, several different alternatives have been implemented in PDEFIT (Schittkowski, 1999):
a) Polynomial approximation: We compute an interpolating polynomial subject to some neighboring values. The number of interpolation points depends on the polynomial degree selected. Polynomials of order 3, 5, 7 or 9 are computed by Newton's formula. By differentiating the interpolation formulae, first and second order approximations are obtained. In case of Neumann boundary conditions, Hermite interpolation is used for being able to exploit known derivative values.
b) Difference Formulae: First and second derivatives can be approximated by difference formulae (Schiesser, 1991). Difference formulae with 3 and 5 points for first derivatives are available, that can be applied recursively to get also 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.
c) Upwind Formulae for Hyperbolic Equations: In case of a scalar hyperbolic equation

$$u_t^i = f_x^i(x, t, u^i, p) \tag{24}$$

with a so-called flux function f, approximation by polynomials or difference formulae might become unstable, especially if non-continuous boundary conditions are supplied to describe for example the propagation of shocks (Schiesser, 1991). Eight upwind formulae are available for solving hyperbolic equations, in particular TVD and high resolution schemes. For more information, see the original literature (Yee, 1985), (Chakravarthy, Osher, 1984a, 1984b, 1985), (Sweby, 1984), (Wang, Richards, 1991), and (Yang, Przekwas, 1992). TVD stands for *Total Variation Diminishing* and the corresponding one parameter family of upwind formulae was proposed by (Chakravarthy, Osher, 1984a). 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.

d) Systems of Advection-Diffusion Equations: Systems of non-homogeneous, nonlinear advection equations

$$u_t^i = f_x^i(u^i, p)_x + g^i(x, t, u^i, u_x^i, u_{xx}^i, p) \tag{25}$$

with area index i, $i = 1, \ldots, n_a$ and $u^i \in I\!R^{n_p}$, $n_p \geq 1$, can be solved by essentially non-oscillatory (ENO) schemes (Harten e. al, 1987), (Harten, 1989), (Walsteijn, 1993). High order polynomials are applied to approximate a 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 w.r.t. u^i is performed, and the scalar ENO method is applied to coefficient functions after a suitable transformation. The resulting system of ordinary differential equations can be solved either by implicit ODE solvers as before, or by a special Runge-Kutta method with fixed stepsizes to satisfy the CLF condition.

Whenever a boundary or transition condition is given in Dirichlet-form, then we know the value of the boundary function and use it to interpolate or approximate the function $u(x, t, p)$ 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. 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.

3.7 Numerical Algorithms for 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 w.r.t. the spatial variable x. It is assumed that always the last n_{ae} equations of the n_p given ones are algebraic. Again 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 $u(x, t, p)$ w.r.t. the spatial variable at a given point x, we may apply polynomial approximation or difference formulae 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 any

sense.

3.8 Statistical Error Analysis

Proceeding from the assumption that the model is sufficiently linear in a neighborhood of an optimal solution 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.

4 Software Organisation and User Interface

EASY-FIT 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, Laplace transforms, steady-state systems, ordinary differential and differential algebraic equations
PDEFIT	parameter estimation in one-dimensional, time-dependent partial differential algebraic equations

EASY-FIT requires a lot of system resources to run in a smooth and efficient way. Recommended hardware are at least 32 MB memory and a fast processor with 400 MHz or more. A full installation requires about 45 MB on hard disk. The system runs under Windows 95, Windows 98, and Windows NT.

EASY-FIT comes with the run-time and royalty-free version of MS-Access. Plots are generated either by internal plot facilities or optionally by the external graphic systems MS-Graph5 or GNUPLOT[1]. All numerical data describing problems and models are kept in a separate database with file name EASY_DAT.MDB to allow easier maintenance and updates.

Parameter estimation problems are solvable without a Fortran compiler, if the model functions are defined in form of the PCOMP modeling language. In this case nonlinear model functions are interpreted and evaluated during run time together with their derivatives. EASY-FIT allows also the most flexible input of the underlying model in form of Fortran code, and has interfaces for the Watcom F77/386[2], the Salford FTN77[3], the Lahey F77L-EM/32, the Digital Visual Fortran, and the Microsoft Fortran PowerStation[4] compiler, where the compiler and linker options can be altered and adapted to special needs.

[1] ©1986-93 by T. Williams, C. Kollin
[2] WATCOM is a registered trademark of WATCOM Systems Inc.
[3] FTN77 is a trademark of Salford Software Ltd.
[4] PowerStation is a trademark of Microsoft Corporation

To install EASY-FIT, one has to insert the CD-ROM and to execute the setup program SETUP.EXE. Installation assumes that the run-time version of MS-Access is to be loaded together with EASY-FIT. More than 550 test examples a included, from which a suitable reference model can be selected. A complete list is found in the EASY-FIT documentation, which comes either in printed form (about 320 pages) or in form of a PDF file..

EASY-FIT is delivered with Fortran source codes of MODFIT.FOR and PDEFIT.FOR, and of corresponding interfaces to the PCOMP interpreter (MODFITEX.FOR and PDEFITEX.FOR). It is possible to link numerical algorithms with user-provided dimensioning parameters or to solve problems, where model functions are defined in form of a Fortran code. All additional object codes of internal numerical algorithms, e.g. for solving the optimization problem or for integrating differential equations, are copied to a separate directory by the setup program, as required by the user.

The graphical user interface of EASY-FIT is menu-driven, where numerical data are inserted into input masks or tables, respectively, and alternative options selected by pull-down lists or select buttons. The available commands allow to define or alter data and functions, to start an optimization run or to get reports on numerical results.

1. File Command

   By selecting either a name from the pick-list or by typing the key word, an available problem in the database is 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. New problems can be generated by the second option of the File command. Another option of the File command is to export or import a problem to or from text files in a format specified by EASY-FIT. Possible reasons for exporting data are e.g. the execution of the numerical codes outside of the database or the possibility to copy problem data to another system. Experimental data can be read from any ASCII-file either in formatted or unformatted form. Moreover, the File command contains a couple of record operations to move through the database, e.g. to go to the first, previous, next or last record. A record corresponds to one parameter estimation problem. Finally the File command allows to define a filter for selecting a subset of parameter estimation problems from the database.

2. Edit Command

   All data that define the dynamical model, can be changed subsequently by moving to the corresponding input field directly or by activating a corresponding sub-form. Model functions are inserted or modified by the internal standard editor of EASY-FIT. The PCOMP parser can be started immediately from the editor form. If Fortran code is preferred, compilation and link is possible directly from the editor form. The order by which variables and functions are to be inserted, is predetermined by the underlying model structure.

3. Start Command

   Depending on the mathematical model type, EASY-FIT executes either MODFIT or PDEFIT, respectively. The codes perform a simulation at a given parameter set or start an optimization cycle. EASY-FIT generates a suitable input file with all data

and tolerances in a format required by the numerical algorithm. After termination, the results are stored in the database. If numerical results are available from a previous run, the user is asked whether he would like to perform a restart, i.e. a simulation or optimization run starting from known parameters.

4. Report Command

   The Report command serves to get either a text report or a function and data plot. The text report is generated directly from the database, and summarizes numerical data and results. Moreover, model functions together with the experimental data, individual residuals, and state variables of partial differential equations are plotted, in the latter case in form of interactive 3D-plots. Alternatively, plots can be generated for MS Graph5 or GNUPLOT.

5. Print Command

   Reports and plots generated by the Report command on screen, are sent to the line printer directly.

6. Copy Command

   All problem data including the model function file, can be copied to another one either within the actual database or to an external EASY-FIT database. In the latter case the database must exist and must possess the same structure as EASY-FIT. Another option of the command is to insert a problem from an external database directly to the actual one. A particular advantage is a more convenient update, if a new EASY-FIT version is to be installed.

7. Delete Command

   The actual problem is deleted from the database. Alternatively a search mask may be defined by the user, to delete a complete subset of parameter estimation problems from the database.

8. Make Command

   If model functions are implemented in form of a Fortran subroutine, the command allows to compile and link the code. The corresponding compiler and linker calls are adapted by the Utilities command.

9. Utilities Command

   Through a couple of menu items EASY-FIT can be adapted to individual situations. The following sub-commands are available:

   (a) Compiler Options: A DOS-batch file with name COMPILE.BAT must contain all necessary compiler options. Some default execution commands for the compilers are included, for which object codes are supplied.

   (b) Linker Options: Similar to the compiler, also all linker options can be adapted and reset by a user.

(c) Dimensioning Parameters: Dimensioning parameters of the codes PDEFIT and MODFIT can be adapted by editing the corresponding include file. The meaning of the parameters is documented by initial comments.

(d) Update Database: After terminating a simulation or optimization run, the numerical results are read from external files only if the actual record is changed or if some other commands are initiated. Alternatively is is possible to require an immediate update of the database.

(e) System Configuration: A few default directory names for exporting or importing files are set, an alternative Windows editor can be selected, the desired graphics software is chosen, and the available Fortran compiler is defined, if needed at all.

(f) Generating Simulated Measurements: Proceeding from a previous simulation run, it is possible to insert the simulation results into the table containing the experimental data. Subsequently the given parameter values can be disturbed and a data fitting run can be started to check the validity of the model, for example whether parameters can be identified uniquely or not.

(g) Adding Randomly Generated Errors to Measurements: In some cases it might be desirable to add some randomly generated errors to the actual measurements, e.g. if exact data of a test example from literature are available and if one wants to add some noise.

Corresponding screen plots are shown in the subsequent section in form of a case study.

5 Case Study: Cooling in a Hot Strip Mill

We consider a mathematical model for cooling a thin plate of thickness L in a rolling mill by water at one side, and by surrounding air at the other side. For simplicity, only one cooling section is considered and we suppose, that the temperature can be measured at both sides of the plate. Moreover, we assume constant speed and neglect heat transfer orthogonal to the move direction. Then we can apply the standard one-dimensional heat equation

$$C_p(T(z,t))p(T(z,t))\frac{\partial T(z,t)}{\partial t} = \frac{\partial}{\partial z}\left(\lambda(T(z,t))\frac{\partial T(z,t)}{\partial z}\right) \quad , \tag{26}$$

where $T(z,t)$ denotes the temperature at time t and the spatial position z orthogonal to the surface of the plate. The density $p(T)$ is given by

$$p(T) = k_p^0 + k_p T \quad ,$$

the heat transfer coefficient $\lambda(T)$ by

$$\lambda(T) = k_\lambda^0 + k_\lambda T \quad ,$$

and the specific heat capacity $C_p(T)$ by piecewise linear interpolation (Groch, 1990), (Kopp, Philipp, 1992), (Chen, 1991), (Ihme, Flaxa, 1991).

Neumann boundary conditions are obtained by combining Newton and Stefan-Boltzmann laws

$$\begin{aligned} \lambda(T(0,t))\frac{\partial T(0,t)}{\partial z} &= \alpha(T(0,t)-T_a)+\epsilon(T(0,t))C(T^4(0,t)-T_a^4) \ , \\ -\lambda(T(L,t))\frac{\partial T(L,t)}{\partial z} &= \alpha(T(L,t)-T_a)+\epsilon(T(L,t))C(T^4(L,t)-T_a^4) \ , \end{aligned} \tag{27}$$

where C denotes the Stefan-Boltzmann constant, i.e. $C = 5.57 \cdot 10^{-8}$, and $\epsilon(T)$ the emission degree

$$\epsilon(T) = T(k_\epsilon T + k_\epsilon^0) + k_\epsilon^1 \ ,$$

(Seredynski, 1973). The boundary conditions are formulated for air cooling at both sides with temperature $T_a = T_{air} = 50$. One side of the plate ($z = 0$) is cooled by water with $T_a = T_{wat} = 20$ in the time interval from $t_1^w = 5$ to $t_2^w = 20$. Initial temperature is set to $T(z, 0) = T_0 = 900$. Note that temperature is given in Celsius, but all other values are normalized.

The partial differential equation is discretized at 50 equidistant spatial intervals and the time integration is performed with RADAU5 (Hairer, Wanner, 1991) with termination accuracy 10^{-4}. The integration is restarted at the two switching points $t_1^w = 4$ and $t_2^w = 20$, and gradients are approximated by forward differences.

We want to investigate the question, whether the corresponding heat transfer constants $\alpha = \alpha_{wat}$ and $\alpha = \alpha_{air}$ can be identified by numerical simulation. Thus we select arbitrary values $\alpha_{wat} = 300$, $\alpha_{air} = 60$ and generate measurements by simulation at 40 equidistant time values $t_i = i$, $i = 1, \ldots 40$ and the two boundary values $z = 0$ and $z = L = 10$. Then we apply the least squares code DFNLP (Schittkowski, 1988) starting from $\alpha_{wat} = 500$, $\alpha_{air} = 50$, to identify these values. The residual is reduced from $0.11 \cdot 10^5$ to $0.32 \cdot 10^{-10}$ in 14 iterations, where α_{wat} and α_{air} are identified subject to 8 correct digits. When adding a randomly generated error of 1% to the simulated data, we obtain the subsequent parameter values listed together with 1% confidence regions:

α_{wat}	283.4	303.9	324.3
α_{air}	51.5	57.1	62.7

The exact parameter values are within the predicted tolerances.

Within EASY-FIT, model equation (26) is defined by the lines

```
*     FUNCTION flux
      lambda = K0lam + Klam*T
      flux = lambda*T_z
C
*     FUNCTION T_t
      P = K0p + Kp*T
      T_t = f_z/(Cp(T)*P)
```

whereas the more complex boundary conditions are given in the form

```
*     FUNCTION T_z_left
      lambda = K0lam + Klam*T
      if ((time.gt.tw1).and.(time.le.tw2)) then
```

```
        alpha = alpha_wat
        T_a = T_wat
      else
        alpha = alpha_air
        T_a = T_air
      endif
      eps = T*(K0eps + Keps*T) + K1eps
      T_z_left = (alpha*(T - T_a) + eps*C*((T+273.15)**4
     /           - (T_a+273.15)**4))/lambda
C
*     FUNCTION T_z_right
      lambda = K0lam + Klam*T
      alpha = alpha_air
      T_a = T_air
      eps = T*(K0eps + Keps*T) + K1eps
      T_z_right = -(alpha*(T - T_a) + eps*C*((T+273.15)**4
     /            - (T_a+273.15)**4))/lambda
```

EASY-FIT proceeds from the order of the function declaration blocks to assign state equations, boundary and initial values, etc. The remaining parameters are either constants, state variables, or the parameter to be fitted. There are no special conventions for variable names, and the linear interpolation of $C_p(T)$ is defined by

```
*     LININT Cp
         0.0   0.68
       780.0   1.1
       790.0   2.8
       840.0   0.72
       880.0   0.7
       920.0   0.6
      1400.0   0.73
```

The remaining constants not defined so far, are set to

```
*     REAL CONSTANT
      Klam = 10
      K0lam = 1.5E+4
      Kp = -0.33
      K0p = 7.85E+3
      Keps = 0.125E-6
      K0eps = -0.38E-3
      K1eps = 1.1
```

The interactive implementation of the model and the identification of unknown parameters is summarized by five steps:

Step 1: Create a new problem in the database, insert some information strings and in particular experimental data, see Figure 1.

Step 2: Choose type of dynamical model (ODE, PDE, ...), define model structure, and set discretization parameters, see Figure 2.

Step 3: Implement model equations and check correct syntax, see Figure 3.

Step 4: Define parameters to be estimated, select least squares solver, set termination tolerances, and start data fitting run, see Figure 4.

Step 5: Check report, especially parameter values, residuals, and state variable, see Figure 5.

Whenever necessary, model equations, data, or tolerances are refined and a data fitting run is repeated, until parameters are identified up to desired precision.

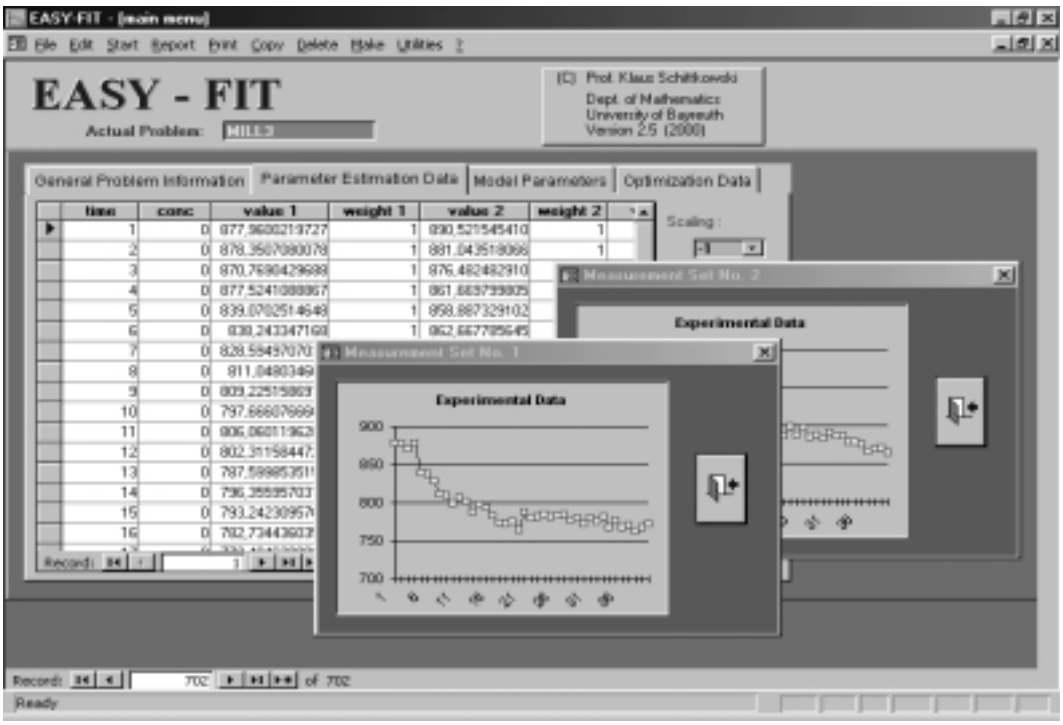

Figure 1: Exponential Data

6 Applications

The demo and the full version of EASY-FIT contain more than 550 *real life* and academic data fitting problems, distributed as follows:

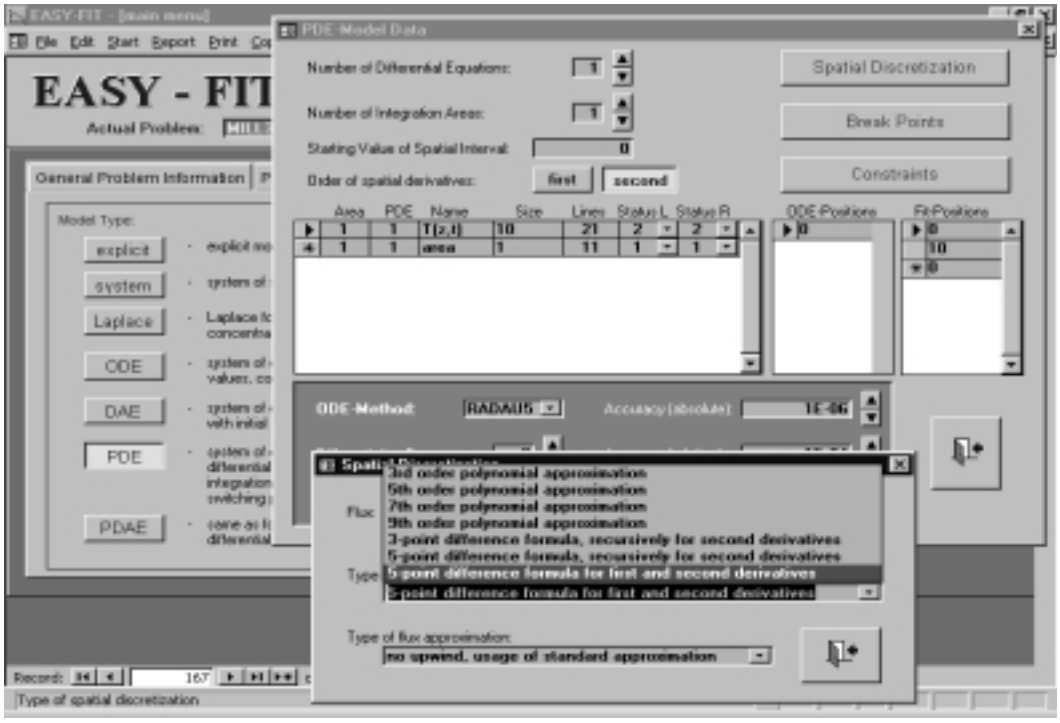

Figure 2: Model Structure and Discretization

explicit model functions	:	69
Laplace transforms	:	8
steady state equations	:	28
ordinary differential equations	:	228
differential algebraic equations	:	22
partial differential equations	:	178
partial differential algebraic equations	:	19
sum	:	552

Most test problems are found in the literature, but there are about 150 test problems derived from cooperative projects with firms or academic institutions applying EASY-FIT. For about 220 test problems there exist experimental data for which an exact solution is not known in advance. The remaining data sets are simulated subject to randomly generated errors. A few practical applications for which published results are available, are listed:

1. Substrate diffusion in a metabolically active cutaneous tissue (Schittkowski, 1998a), (Boderke, Schittkowski, Wolf, Merkle, 1998), (Steingässer, 1994), also formulated as an optimal control model (Blatt, Schittkowski, 1998).

2. Simulation of an acetylene reactor (Wansbrough, 1985), optimal control with online adoption of maintenance intervals (Birk, Liepelt, Schittkowski, Vogel, 1998).

3. Drying of maltodextrin a convection oven (Frias, Oliveira, Schittkowski, 1998).

Figure 3: Dynamical Equations

4. Receptor-ligand binding study (Rominger, Albert, 1985), (Schittkowski, 1994).
5. Multibody system of a truck (Simeon, Grupp, Führer, Rentrop, 1994), (Simeon, Rentrop, 1993), (Führer, Leimkuhler, 1991).
6. Manutec r3 robot (Otter, Türk, 1988).
7. Mass transfer in sorbing porous media (Hoch, 1995), (Van Genuchten, Wierenga, 1976), (Andersson, Olsson, 1985).
8. Ammonium and nitrate fertilization in forest soils (Fischer, 1996).
9. Distillation column (Jourdan, 1997), (Kuhn, Schmidt, 1987).
10. Cooling during rolling-mill operations (Hedrich, 1996), (Groch, 1990).
11. Signal evaluation in periodic mechanical systems (Zschieschang, Rockhausen, 1996).
12. Isomerization in protein unfolding and refolding (Odefey, Mayr, Schmid, 1995), (Mayr, Odefey, Schutkowski, Schmid, 1996).
13. Themodynamic coupling for GroEL-mediated unfolding (Walter, Lorimer, Schmid, 1996).

Moreover EASY-FIT is running at a couple of academic and commercial institutions, for example at BASF, Bayer, Boehringer Ingelheim, Eurocopter, Novartis.

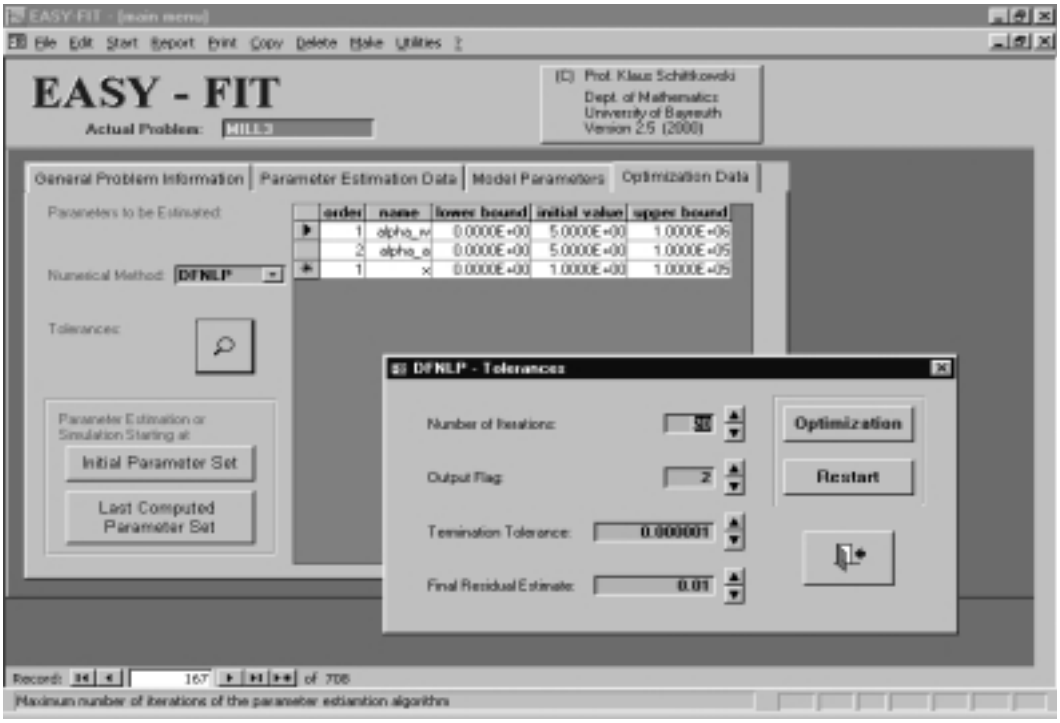

Figure 4: Starting Data Fitting

7 Summary

We introduced a software system to estimate unknown parameters in explicit model functions, steady-state systems, 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.

The general structure of the mathematical models that can be treated, is described in detail together with a brief review on discretization techniques and numerical algorithms. The graphical user interface is outlined and illustrated in form of a case study, showing also the numerical efficiency of the approach.

There are a couple of possible enhancements of the software in future, we are considering at present, for example

1. optimal experimental design, e.g. position of experimental time values,
2. analytical or semi-analytical derivatives of PDE solutions,
3. understanding higher index equations in PDAE's,
4. implicit differential equations,

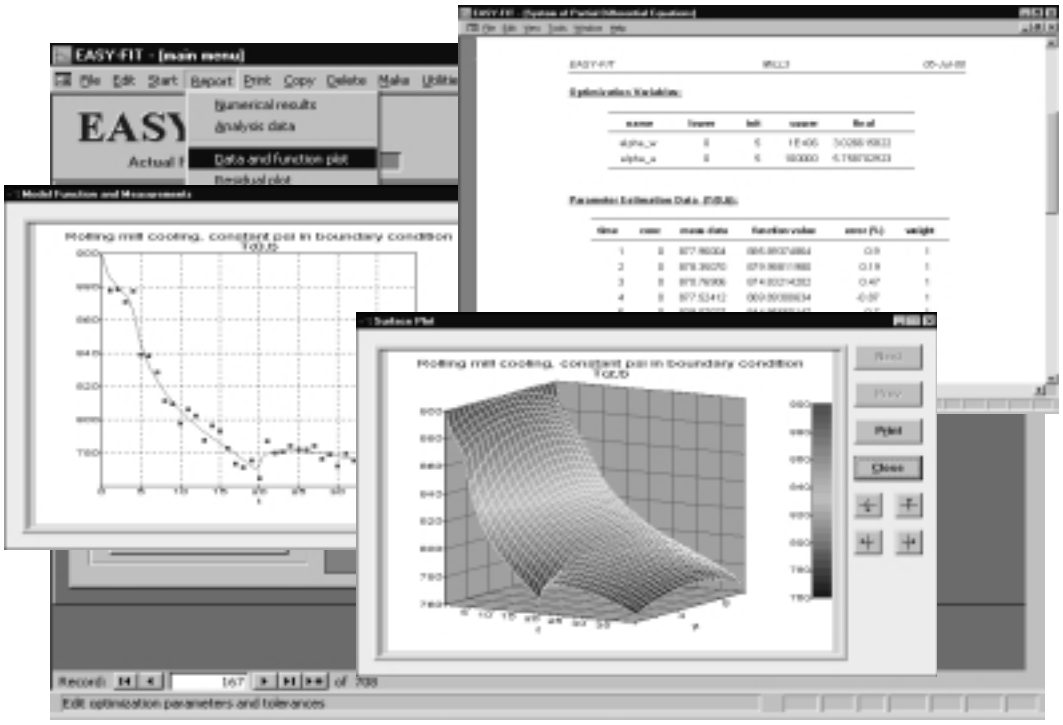

Figure 5: Interpretation of Results

5. more flexible interactive plots,

6. interfaces to office programs (Excel, Word).

For more information about EASY-FIT and its capabilities, contact the author under

klaus.schittkowski@uni-bayreuth.de

A demo version is available through the URL

http://www.klaus-schittkowski.de

References

[1] Andersson F., Olsson B. eds. (1985): *Lake Gådsjön. An acid forest lake and its catchment*, Ecological Bulletins, Vol. 37, Stockholm

[2] Ascher U.M., Petzold L.R. (1998): *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations,* SIAM, Philadelphia

[3] Bellman R.E., Kalaba R.E., Lockett J. (1966): *Numerical Inversion of the Laplace Transform,* American Elsevier, New York

[4] Benecke C. (1993): *Interne numerische Differentiation von gewöhnlichen Differentialgleichungen,* Diploma Thesis, Department of Mathematics, University of Bayreuth

[5] Birk J., Liepelt M., Schittkowski K., Vogel F. (1998): *Computation of optimal feed rates and operation intervals for turbular reactors*, Journal of Process Control, Vol. 9, pp. 325-336

[6] Blatt M., Schittkowski K. (1998): *Optimal Control of One-Dimensional Partial Differential Equations Applied to Transdermal Diffusion of Substrates*, in: Optimization Techniques and Applications, L. Caccetta, K.L. Teo, P.F. Siew, Y.H. Leung, L.S. Jennings, V. Rehbock eds., School of Mathematics and Statistics, Curtin University of Technology, Perth, Australia, Volume 1, pp. 81-93

[7] 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, pp. 95-121

[8] Boderke P., Schittkowski K., Wolf M., Merkle H.P. (1998): *A mathematical model for diffusion and concurrent metabolism in metabolically active tissue*, submitted for publication

[9] Chakravarthy S.R., Osher S.(1984): *High resolution schemes and the entropy condition,* SIAM Journal on Numerical Analysis, Vol. 21, No. 5, pp. 955-984

[10] Chakravarthy S.R., Osher S. (1984): *Very high order accurate TVD schemes,* ICASE Report No. 84-44

[11] Chakravarthy S.R., Osher S. (1985): *Computing with high resolution upwind schemes for hyperbolic equations,* Lectures in Applied Mathematics, Vol. 22, pp. 57-86

[12] Chen J. (1991): *Abkühlungsvorgänge von Stahlplatten mit Spritzwasserbeaufschlagung*, Umformtechnische Schriften, Vol. 30

[13] 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, pp. 369-383

[14] 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, pp. 233-266

[15] Fischer P. (1996): *Modellierung und Simulation der Ammonium- und Nitrat-Dynamik in strukturierten Waldböden unter besonderer Berücksichtigung eines dynamischen, hierarchischen Wurzelsystems,* Diploma Thesis, Department of Mathematics, University of Bayreuth

[16] Frias J.M., Oliveira J.C, Schittkowski K. (1998): *Modelling of maltodextrin DE12 drying process in a convection oven*, submitted for publication

[17] Führer C., Leimkuhler B. (1991): *Numerical solution of differential-algebraic equations for constrained mechanical motion,* Numerische Mathematik, Vol. 59, pp. 55-69

[18] Groch A.G. (1990): *Autmatic control of laminar flow cooling in continuous and reversing hot strip mills,* Iron and Steel Engineer, pp. 16-20

[19] Hairer E., Wanner G. (1991): *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems,* Springer Series Computational Mathematics, Vol. 14, Springer

[20] Hairer E., Nørsett S.P., Wanner G. (1993): *Solving Ordinary Differential Equations I: Nonstiff Problems,* Springer Series Computational Mathematics, Vol. 8, Springer

[21] 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, pp. 231-303

[22] Harten A. (1989): *ENO schemes with subcell resolution,* Journal on Computational Physics, Vol. 83, pp. 148-184

[23] Hartwanger C., Schittkowski K. (1999): *Computer aided design of horn radiators for satellite communication by least squares optimization*, to appear: Engineering Optimization

[24] Hedrich C. (1996): *Modellierung, Simulation und Parameterschätzung von Kühlprozessen in Walzstraßen*, Diploma Thesis, Department of Mathematics, University of Bayreuth

[25] Hoch R. (1995): *Modellierung von Fließwegen und Verweilzeiten in einem Einzugsgebiet unter stationären Fließbedingungen,* Diplomarbeit, Fakultät für Biologie, Chemie und Geowissenschaften, Universität Bayreuth

[26] Ihme F., Flaxa V. (1991): *Intensivkühlung von Fein- und Mittelstahl*, Stahl und Eisen, Vol. 112, pp. 75-81

[27] Jourdan, M. (1997): *Simulation und Parameteridentifikation von Destillationskolonnen*, Diploma Thesis, Department of Mathematics, University of Bayreuth

[28] Kopp R., Philipp F.D. (1992): *Physical parameters and boundary conditions for the numerical simulation of hot forming processes*, Steel Research, Vol. 63, pp. 392-398

[29] 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, pp. 75-95

[30] 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, No. 17, pp. 5550-5561

[31] Nelder J.A., Mead R. (1965): *A simplex method for function minimization,* The Computer Journal, Vol. 7, p. 308

[32] Nickel B. (1995): *Parameterschätzung basierend auf der Levenberg-Marquardt-Methode in Kombination mit direkter Suche,* Diploma Thesis, Department of Mathematics, University of Bayreuth

[33] 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, pp. 69-78

[34] Otter M., Türk S. (1988): *The DFVLR models 1 and 2 of the Manutec r3 robot,* DFVLR-Mitt. 88-3, DFVLR, Oberpfaffenhofen

[35] Prince P.J., Dormand J.R. (1981): *High order embedded Runge-Kutta formulae,* Journal on Computational Applied Mathematics, Vol. 7, pp. 67-75

[36] Rominger K.L., Albert H.J. (1985): *Radioimmunological determination of Fenoterol. Part I: Theoretical fundamentals,* Arzneimittel-Forschung/Drug Research, Vol.35, No.1a, pp. 415-420

[37] Schiesser W.E. (1991): *The Numerical Method of Lines,* Academic Press, San Diego

[38] Schittkowski K. (1980): *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer

[39] Schittkowski K. (1985/86): *NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, pp. 485-500

[40] 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, pp. 295-309

[41] Schittkowski K. (1994): *Parameter estimation in systems of nonlinear equations,* Numerische Mathematik, Vol. 68, pp. 129-142

[42] Schittkowski K. (1997): *Parameter estimation in partial differential equations,* Optimization Methods and Software, Vol. 7, No. 3-4, pp. 165-210

[43] Schittkowski K. (1998): *Parameter estimation in a mathematical model for substrate diffusion in a metabolically active cutaneous tissue*, to appear: Progress in Optimization II, pp. 183-204

[44] Schittkowski K. (1999): *PDEFIT: A FORTRAN code for parameter estimation in partial differential equations*, Optimization Methods and Software, Vol. 10, pp. 539-582

[45] Seredynski F. (1973): *Prediction of plate cooling during rolling mill operation*, Journal of the Iron and Steel Institute, Vol. 211, pp. 197-203

[46] Shampine L.F., Watts H.A. (1979): *The art of writing a Runge-Kutta code,* Applied Mathematics and Computations, Vol. 5, pp. 93-121

[47] 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, pp. 469-474

[48] 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, pp. 523-532

[49] Stehfest H. (1970): *Algorithm 368: Numerical inversion of Laplace transforms,* Communications of the ACM, Vol. 13, pp. 47-49

[50] Steingässer I. (1994): *The organized HaCaT cell culture sheet: A model approach to study epidermal peptide drug metabolism,* Dissertation, Pharmaceutical Institute, ETH Zürich

[51] Sweby P.K. (1984): *High resolution schemes using flux limiters for hyperbolic conservation laws,* SIAM Journal on Numerical Analysis, Vol. 21, No. 5, pp. 995-1011

[52] 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, pp. 892-898

[53] 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

[54] Walter S., Lorimer G.H., Schmid F.X. (1996): *A thermodynamic coupling mechanism for GroEl-mediated unfolding*, Biochemistry, Vol. 93, pp. 9425-9430

[55] Wang Z., Richards B.E. (1991): *High resolution schemes for steady flow computation,* Journal of Computational Physics, Vol. 97, pp. 53-72

[56] Wansbrough R.W. (1985):*Modeling chemical reactors*, Chemical Engineering, Vol. 5, pp. 95-102

[57] 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, pp. 139-159

[58] Yee H.C. (1985): *Construction of a class of symmetric TVD schemes,* Lectures in Applied Mathematics, Vol. 22, pp. 381-395

[59] Zschieschang T., Rockhausen L. (1996): *Zur Signalauswertung und Modellierung bei periodischen Vorgängen mit transienten Anteilen*, Report, Institute of Mechanics, Technical University of Chemnitz-Zwickau, Chemnitz, Germany