
Optimization Methods and Software

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/goms20

# Numerical tools for parameter estimation in ode-systems

Lennart Edsberg [a] & Per-Åke Wedin [b]

[a] NADA, KTH, Stockholm, S-100 44, Sweden E-mail:

[b] Department of Computing Science, Umeå University, Umeå, S-901 87, Sweden E-mail:

Version of record first published: 22 Dec 2010.

To cite this article: Lennart Edsberg & Per-Åke Wedin (1995): Numerical tools for parameter estimation in ode-systems, Optimization Methods and Software, 6:3, 193-217

To link to this article: http://dx.doi.org/10.1080/10556789508805633

*Optimization Methods and Software*, 1995, Vol. 6, pp. 193–217
Reprints available directly from the publisher
Photocopying permitted by license only



# NUMERICAL TOOLS FOR PARAMETER ESTIMATION IN ODE-SYSTEMS

LENNART EDSBERG and PER-ÅKE WEDIN

*NADA, KTH, S-100 44 Stockholm, Sweden*
*Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden*
*e-mail edsberg@nada.kth.se, pwedin@cs.umu.se*

*(Received 13 December 1993; in final form 3 July 1995)*

The numerical problem of estimating unknown parameters in systems of ordinary differential equations from complete or incomplete data is treated. A new numerical method for the optimization part, based on the Gauss-Newton method with a trust region approach to subspace minimization for the weighted nonlinear least squares problem, is presented. The method is implemented in the framework of a toolbox (called diffpar) in Matlab and several test problems from applications, giving non-stiff and stiff ODE-systems, are treated.

KEY WORDS: Parameter estimation, ODE-systems, weighted nonlinear least squares



## 1 INTRODUCTION

The problem of estimating parameters in systems of ordinary differential equations occurs in different dynamical applications, such as chemical kinetics and control theory. The problem has mathematical, numerical and statistical aspects. This paper has three purposes. The first purpose is to give an overview of mathematical formulations and numerical methods for solving the problem. The second purpose is to give a description of a new algorithm for regularization of least squares problems and the third purpose is to present a toolbox, written in Matlab, for computation of parameters in systems of ordinary differential equations from given data. The toolbox, called *diffpar* (*diff*erential equations with unknown *par*ameters), can be used to perform numerical experiments and to solve practical problems within this problem class.

*Diffpar* is based on a Gauss-Newton type algorithm for minimizing an objective function which is the square of a weighted norm of residuals:

$$\phi(\Theta) = \frac{1}{2}\sum_{i=0}^{m}(\mathbf{y}(t_i, \Theta) - \tilde{\mathbf{y}}_i)^T \mathbf{W}_i (\mathbf{y}(t_i, \Theta) - \tilde{\mathbf{y}}_i) \tag{1}$$

where $\mathbf{y}(t, \Theta)$ satisfies the system of ordinary differential equations

$$\dot{\mathbf{y}}(t, \Theta) = \mathbf{f}(t, \mathbf{y}(t, \Theta), \mathbf{k}),\ \mathbf{y}(t_0) = \mathbf{y}_0 \tag{2}$$

In (1) and (2) $\Theta$ is the parameter vector defined as $\Theta^T = (\mathbf{k}^T, \mathbf{y}_0^T)$ and $(t_i, \tilde{\mathbf{y}}_i)$, $i = 0, 1, \ldots, m$ are the measurements of the state vector $\mathbf{y}(t, \Theta)$. The matrices $\mathbf{W}_i$, $i = 0, 1, \ldots, m$ are diagonal weight matrices, with nonnegative elements. In *diffpar*, the system of ordinary differential equations is solved with a discretization method combined with dense output interpolation, giving the numerical solution $\mathbf{y}_i(\Theta)$, $i = 0, 1, \ldots, m$ at the $t_i$-points.

The formulation of the parameter estimation problem as given above is a *nonlinear weighted least squares problem.*

Let us illustrate the problem of estimating parameters in a system of ordinary differential equations by considering an example based on a mathematical model occurring in several applications, such as chemical kinetics, theoretical biology, ecology, etc. This example is known as Barnes' problem and is based on the Lotka-Volterra differential equations which define a system of $n = 2$ differential equations formulated as an initial value problem:

$$\frac{dy_1}{dt} = k_1 y_1 - k_2 y_1 y_2, \quad y_1(0) = a \tag{3}$$

$$\frac{dy_2}{dt} = k_2 y_1 y_2 - k_3 y_2, \quad y_2(0) = b \tag{4}$$

If we want to estimate the rate constants $k_1, k_2, k_3$ and the initial values $a, b$ this problem has $p = 5$ unknown parameters. The following 11 measurements ($m = 10$) are given:

TABLE 1: Measurements of an oscillating process.

| $t$ | 0.00 | 0.50 | 1.00 | 1.50 | 2.00 | 2.50 | 3.00 | 3.50 | 4.00 | 4.50 | 5.00 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tilde{y}_1$ | 1.00 | 1.10 | 1.30 | 1.10 | 0.90 | 0.70 | 0.50 | 0.60 | 0.70 | 0.80 | 1.00 |
| $\tilde{y}_2$ | 0.30 | 0.35 | 0.40 | 0.50 | 0.50 | 0.40 | 0.30 | 0.25 | 0.25 | 0.30 | 0.35 |

To illustrate some of the possibilities when using *diffpar*, we show how parameters can be estimated under different conditions:

(a) The initial values $a$ and $b$ are known and the measurements are used to estimate $k_1$, $k_2$ and $k_3$.

(b) All parameters, initial values $a$, $b$ and rate constants $k_1$, $k_2$, $k_3$ are to be estimated from the measurements.

(c) Different weights are applied to the measurements.

(d) Estimate $k_1$, $k_2$ and $k_3$ when only one of the components $y_1$ or $y_2$ can be measured.

We show here the outcome of the estimation of parameters under condition (a). The solutions of the problem under the other conditions are referred to section 4.

For a nonlinear least squares problem a start vector $\Theta_0$ must be given for the parameters to be estimated. Thus for problem (a) the user of *diffpar* needs to "guess" $k_1, k_2, k_3$, while the initial values are assumed to be known: $a = 1.0, b = 0.3$. In the first of the two graphs below we see the measurement points (x) and the solution of the system of ordinary differential equations (solid lines) for the starting value $\Theta_0 = (1, 1, 1)^T$, for which $\phi(\Theta_0) = 8.625$ ($\mathbf{W}_i = \mathbf{I}$, $i = 0, 1, \ldots, m$).

After a few iterations with *diffpar* we see the result in the second graph. The discrepancies between the model and the data have been reduced substantially. For the estimate $\Theta_{\text{est}}^{(1)} = (0.8609, 2.0787, 1.8147)^T$ of the parameters we obtain a local minimum $\phi(\Theta_{\text{est}}^{(1)}) = 0.0823$. Although the objective function has a small value for $\Theta = \Theta_{\text{est}}^{(1)}$ the fit is not so good everywhere. This can be adjusted and examined interactively by giving different weights to the measurement points, see section 4.

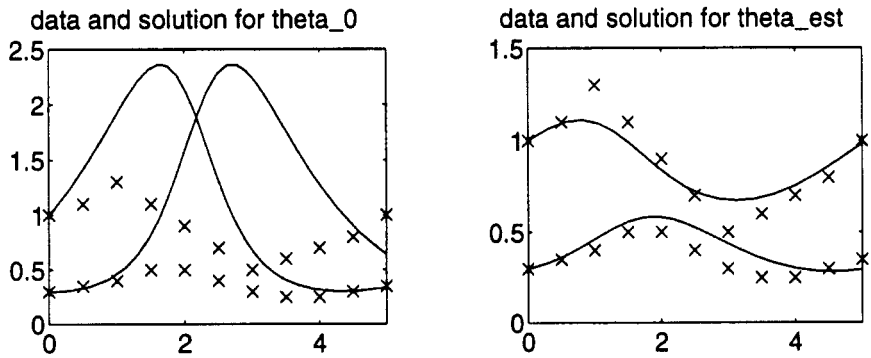


FIGURE: 1

However, with another starting vector $\Theta_0 = (3, 3, 3)^T$ the outcome is not successful. The estimate is computed to $\Theta_{\text{est}}^{(2)} = (1.66, 5.24, 4.89)^T$, for which the objective function has a local minimum $\phi(\Theta_{\text{est}}^{(2)}) = 0.39$. The start vector $\Theta_0 = (0.3, 0.3, 0.3)^T$ is even worse, since it converges to $\Theta_{\text{est}}^{(3)} = (-0.60, -2.22, -1.94)^T$, a parameter vector with negative rate constants. In *diffpar* it is possible to iterate in the logarithmated parameters $\Psi = \ln(\Theta)$. With that option the $\Theta$-values are forced to be positive. Use of logarithmated parameters does not affect the computational efficiency of *diffpar*.

The rest of the paper is organized as follows: In section 2 we give the general problem formulation and show two different discretization methods, one leading to a constrained formulation and the other one to an unconstrained formulation of the nonlinear least squares problem. In section 3 we focus on the unconstrained formulation and describe algorithms used in *diffpar*. We use a new variant for regularization of Gauss-Newton's algorithm based on a trust region method for subspace minimization according to Wedin and Edsberg [31] and discretization methods for solving the system of ordinary differential equations, stiff or nonstiff. In section 4 we illustrate with examples from different applications and report results from the use of *diffpar*.

## 2 PROBLEM FORMULATION AND DISCRETIZATIONS

### 2.1 *Analytical formulation*

Assume we have a mathematical model of a dynamical process consisting of a system of $n$ ordinary differential equations (ODEs) with a parameter vector $\mathbf{k}$ of dimension $p$ in the right hand sides:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}, \mathbf{k}),\ \mathbf{y}(t_0) = \mathbf{y}_0 \tag{5}$$

We want to fit this model to data, which consist of a point set of given measurements: $(t_i, \tilde{\mathbf{y}}_i)$, $i = 0, 1, 2, \ldots, m$.

The parameters $\Theta$ of the model are defined by the vector $\Theta^T = (\mathbf{k}^T, \mathbf{y}_0^T)$ having dimension $s = p + n$. The notation $\mathbf{k}$ for the parameters in the right hand side of the ODEs is an influence from chemical kinetics, where the rate constants of the reactions are often denoted $k_1, k_2$, etc.

Common ways of defining the parameter estimation problem lead to a constrained nonlinear weighted least squares problem, the analytical formulation of which is:

Define the object function $\phi(\Theta, \Upsilon)$:

$$\phi(\Theta, \Upsilon) = \frac{1}{2} \sum_{i=0}^{m} (\mathbf{y}(t_i, \Theta) - \tilde{\mathbf{y}}_i)^T \mathbf{W}_i (\mathbf{y}(t_i, \Theta) - \tilde{\mathbf{y}}_i) \tag{6}$$

and formulate the constrained problem

$$\min_{(\Theta, \Upsilon)} \phi(\Theta, \Upsilon) \tag{7}$$

$$\text{s.t. } \dot{\mathbf{y}}(t_i, \Theta) = \mathbf{f}(t_i, \mathbf{y}(t_i, \Theta), \mathbf{k}),\ \mathbf{y}(t_0) = \mathbf{y}_0 \tag{8}$$

where $\Upsilon$ in (6) and (7) is the compound vector $\Upsilon^T = (\mathbf{y}(t_1, \Theta)^T, \ldots, \mathbf{y}(t_m, \Theta)^T)$ and $\mathbf{W}_i$ in (6) are diagonal weight matrices with nonnegative elements.

Since the solution of the ODE-system cannot be given analytically, in general, the differential constraints in (8) must be discretized somehow to give a number of algebraic constraints.

### 2.2 *Numerical formulations*

*2.2.1 Constrained formulation.* Discretized formulations of the problem can be given in different ways. Difference approximation of the ODE-system leads to a constrained formulation:

$$\min_{(\Theta, \mathbf{Y})} \frac{1}{2} \sum_{i=0}^{m} (\mathbf{y}_i - \tilde{\mathbf{y}}_i)^T \mathbf{W}_i (\mathbf{y}_i - \tilde{\mathbf{y}}_i) \tag{9}$$

$$\text{s.t. } \mathbf{D}_h \mathbf{Y} - \mathbf{F}(\mathbf{Y}, \mathbf{k}) = 0 \tag{10}$$

where $\mathbf{Y}^T = (\mathbf{y}_1^T, \ldots, \mathbf{y}_m^T)$ is the numerical solution of (2) according to a discretization, leading to the algebraic constraints in (10). $\mathbf{D}_h$ is a matrix corresponding to the discretization of $\dot{\mathbf{y}}$ and $\mathbf{F}$ is a nonlinear function of $\mathbf{Y}$ and $\mathbf{k}$, the form of which depends on the discretization method that has been used.

When using an explicit Runge-Kutta method or an implicit BDF-method with a stepsize sequence of $N$ steps in the discretization of the $t$-variable: $t_0, t_1, \ldots t_N$, there will be $nN$ constraints (10) for the $nN + s$ variables, thus leaving $s$ degrees of freedom to fit the parameter vector $\boldsymbol{\Theta}$.

Methods for parameter estimation in ODE's based on a constrained formulation has been used by e.g. Tjoa and Biegler [29]. In their approach the solution of the ODE-system is approximated by a collocation method as piecewise polynomials in the interval of integration, as developed by Baden and Villadsen [3]. When using orthogonal collocation based on Legendre polynomials this is equivalent to using an implicit Runge-Kutta method. Hence using this method, the ODE's are approximated by a large number of algebraic equations being the constraints of (10) and a large number of variables to be minimized with respect to. The resulting nonlinear least squares problem with nonlinear constraints is solved with sequential quadratic programming in Biegler's approach. The sparsity structure is utilized to handle the size of the problem effectively. One difficulty with this approach is to find a start vector for the augmented parameter vector $(\boldsymbol{\Theta}, \mathbf{Y})$. For the $\mathbf{Y}$-part of this vector, the given data can be interpolated to give start values. However, if measurements are not available for the whole state vector $\mathbf{y}$, there will be some components for which we have to guess also the corresponding measurement values.

*2.2.2 Unconstrained formulation.* When the $\mathbf{y}_i$-variables in the constraints (10) are solved in terms of $\mathbf{k}$ and $\mathbf{y}_0$, we get an unconstrained formulation:

$$\min_{\boldsymbol{\Theta}} \frac{1}{2} \sum_{i=0}^{m} (\mathbf{y}_i(\Theta) - \tilde{\mathbf{y}}_i)^T \mathbf{W}_i (\mathbf{y}_i(\Theta) - \tilde{\mathbf{y}}_i) \tag{11}$$

where $\mathbf{y}_i(\boldsymbol{\Theta})$, $i = 1, 2, \ldots, m$ is the numerical solution of (2) for a given value of $\boldsymbol{\Theta}$. In the formulation (11) we minimize with respect to the unknown parameter vector $\boldsymbol{\Theta}$ only, i.e. the degree of freedom is $s = \dim(\boldsymbol{\Theta})$.

The unconstrained formulation of parameter estimation in ODE-systems has since long been the classical approach to the problem. One of the earliest references in this area is found in Rosenbrock and Storey [26], where chapter 8 is devoted to the algorithmic principles of this problem applied to estimation of rate constants in chemical reactions. Another classical reference of this area is Bard [4], chapter 7, where several problems from chemical kinetics are treated numerically and statistically.

Several attempts have been made to construct program tools for parameter estimation in ODE's. One of the first, FACSIMILE by Curtis *et al.* [7], was available in the early 70's and was designed for rate constant estimation in kinetic problems. One of us had the privilege, at a visit to Harwell in 1972, of testing one of the first versions of that program tool on a problem known as Robertson's problem, see section 4.2. The results of that test were published as a technical report in Curtis and Edsberg [8]. The numerical software in FACSIMILE is based on Harwell subroutines for ODE-solution and optimization. Another

early software package, published by Domselaar and Hemker [10] in 1975, which has lately been updated by Hemker and Kok [18], is based on a stiff ODE-solver and Marquardt's method. In that report there is also described a method for computing a start vector $\boldsymbol{\Theta}_0$ for the iterations, based on multiple shooting. Recent contributions within the area is reported by Schittkowski [28]. Other relevant publications connected to this problem is found in Ekstam and Smed [12], Williams [33], Leis and Kramer [20] and Aiken [1].

*2.2.3 Mixed formulation.* In the early 80's a group at Heidelberg University led by Deuflhard published algorithms and program tools for parameter estimation in ODE's, see Nowak and Deuflhard [23]. Bock [6] gives a description of a program based on an ODE-method and a modified Gauss-Newton method by Deuflhard [23]. They report successful runs on large problems from rate constant estimation in chemical reactions. Bock's approach is based on a multiple shooting technique for solving a constrained overdetermined multipoint boundary value problem:

In addition to the parameters $\boldsymbol{\Theta}$, the unknown state vectors $\bar{\mathbf{y}}_q$ at the nodes $\tau_q$, $t_0 = \tau_0 < \tau_1 < \ldots \tau_Q = t_m$, of a sufficiently fine grid on $[t_0, t_m]$ are chosen as variables. For a given iterate of the augmented parameter vector $(\boldsymbol{\Theta}, \bar{\mathbf{Y}})$, where $\bar{\mathbf{Y}}^T = (\bar{\mathbf{y}}_0^T, \bar{\mathbf{y}}_1^T, \ldots, \bar{\mathbf{y}}_{Q-1}^T)$, the $Q$ initial value problems

$$\dot{\mathbf{y}}^{(q)} = \mathbf{f}(t, \mathbf{y}^{(q)}, \mathbf{k}),\ \mathbf{y}^{(q)}(\tau_q) = \bar{\mathbf{y}}_q,\ q = 0, 1, \ldots, Q-1 \tag{12}$$

are solved, which gives a discontinuous trajectory on $[t_0, t_m]$. By adding the matching conditions

$$\mathbf{y}^{(q)}(\tau_{q+1}) - \bar{\mathbf{y}}_{q+1} = 0,\ q = 0, 1, \ldots, Q-1 \tag{13}$$

as constraints we get a variant of the constrained formulation which in our notation can be formulated as

$$\min_{(\boldsymbol{\Theta}, \bar{\mathbf{Y}})} \frac{1}{2} \sum_{i=0}^{m} (\mathbf{y}_i(\Theta) - \tilde{\mathbf{y}}_i)^T \mathbf{W}_i (\mathbf{y}_i(\Theta) - \tilde{\mathbf{y}}_i) \tag{14}$$

$$\text{s.t. } \mathbf{y}^{(q)}(\tau_{q+1}) - \bar{\mathbf{y}}_{q+1} = 0,\ q = 0, 1, \ldots, Q-1 \tag{15}$$

where $\mathbf{y}_i(\boldsymbol{\Theta})$ is the numerical solution of the initial value problem (12) at $t = t_i$.

Note that in (9) and (11) we obtain continuous representations of the ODE-solution in each iterate, while in (14) we will have discontinuous ODE-solutions in the iterations until the problem has converged.

## *2.3 Overview of the methods used in the diffpar toolbox*

In *diffpar* the parameter estimation algorithm is based on the unconstrained formulation (11). In this approach we need to solve the initial value problem on $[t_0, t_m]$ for a set of $\boldsymbol{\Theta}$-values generated by the iteration method. For that reason we call the corresponding parameter estimation algorithm the *initial value approach.*

The choice of object function (11) admits treatment of experimental situations where not all components of the state vector $\mathbf{y}$ can be measured, but only a few of the components of $\mathbf{y}$. In this case we only put zero weights in the diagonal elements corresponding to components which have not been measured. Of course we also have the possibility of choosing different non-zero weights according to different accuracy of the components of the system, based on e.g. statistical information.

In *diffpar* the numerical algorithm for computing the parameter vector $\Theta$ is based on a Gauss-Newton type method for finding a local minimum of (11) given a starting value $\Theta_0$ of the parameter vector. The algorithm is globally convergent, i.e. converges to a local minimum from an arbitrary starting value of the parameter vector, but there is no guarantee that the global minimum is reached, see also 3.2.1. The Gauss-Newton algorithm is described in detail in section 3.2. We use a new regularization method which is superior to the ordinary trust region approach for problems that are almost singular at the solution. This method uses a trust region approach to subspace minimization and unlike a pseudorank approach it is globally convergent.

In each iteration we solve the initial value problem (2) and the corresponding variational equations numerically. In *diffpar* these algorithms are based on a third order Runge-Kutta method for non-stiff problems or a third order BDF-method for stiff problems. These algorithms are described in section 3.3.

A major problem in nonlinear parameter estimation is to find a starting value $\Theta_0$ for the Gauss-Newton method which makes the iterations converge to the global minimum attained for $\Theta = \Theta^*$. For problems where the parameter dependence at $\Theta^*$ is nearly linear, the starting value can be far away from the optimum and we still have convergence to the correct solution. For problems behaving nonlinearly at $\Theta^*$, i.e. problems with large curvature in the residual space, a very good starting value is needed in order to have convergence to the wanted minimum. The main difficulty with least squares problems that are very nonlinear in $\Theta$ is not the slow convergence close to the solution — a difficulty that can be overcome in lots of ways — but the existence of several local minima and saddle points and the risk of ending up at the wrong minimum. For that reason a global optimization method would be ideal. Such methods are still too slow or unsatisfactorily investigated. The next best thing to do is to have a good starting point in the vicinity of the solution. As already pointed out in 2.2.2 the shooting method is a good method for obtaining starting values of the parameters, but measurements from all components $y_i$ must be available. For some problem classes the structure of the problem can be utilized; in many chemical kinetic problems the parameters enter linearly in the right hand side of the ODE-system, i.e. $\mathbf{f}(t, \mathbf{y}, \mathbf{k}) = \mathbf{G}(t, \mathbf{y})\mathbf{k}$, where $\mathbf{G}$ is a nonlinear matrix-valued function in $t$ and $\mathbf{y}$. A parameter estimation algorithm based on an integral formulation of the ODE-system (2) can be used to compute a start value for the parameter vector, see Wikström [32].

## 3 A NUMERICAL METHOD BASED ON THE INITIAL VALUE APPROACH

### *3.1 General discussion of the algorithm*

A straightforward method for parameter estimation in ODE's is based on an optimization routine combined with an ODE-solver. In such an approach, however, it is important

that the optimization routine utilizes the fact that the object function is evaluated by solving an ODE-system numerically, which means that care should be taken regarding the interaction of the numerical mechanisms involved in the two processes. Two important mechanisms in the synthesis of these two processes are the *steplength* determination in the optimization algorithm and the *stepsize* control in the ODE-solver. In the optimization phase the steplength algorithm can produce iterates of the parameter vector with e.g. components outside the stability area of the ODE-system, while different stepsize sequences generated by the automatic stepsize control in the ODE-solver produce a nonsmooth object function to be minimized, which has been treated by e.g. Gear [13], and Hairer, Nørsett and Wanner [16].

For the ODE-system (2), the stepsize sequence $[\mathbf{h}] = h_0, h_1, \ldots, h_{N-1}$ generated by the automatic stepsize control, depends on the parameter vector $\boldsymbol{\Theta}$ and the tolerance parameter *tol* given to the ODE-solver. Each time the Gauss-Newton algorithm computes a new iterate $\boldsymbol{\Theta}_j$, the stepsize sequence $[\mathbf{h}]$ will be changed by the automatic stepsize control. However, from the viewpoint of the optimizer, a given stepsize sequence corresponds to a given overdetermined nonlinear system of algebraic equations. If the stepsize sequence is changed, the algebraic system will be different e.g. the nonlinear equations will change, so the $\boldsymbol{\Theta}_j$-iterates will emerge from different objective functions. Thus the objective function $\phi(\boldsymbol{\Theta})$ will be nonsmooth, and the behaviour of the minimization routine can be very irregular.

In our approach, we have therefore chosen to keep the stepsize sequence fixed as long as the estimated error in the ODE-solution is smaller than $\omega \cdot tol$, where $\omega$ is a tuning factor, say in the interval $1 \leq \omega \leq 10$. In that way the stepsize sequence is kept fixed for a sequence of $\Theta_j$-iterates, hence not changing the objective function $\phi(\Theta)$. This is possible as long as the $\boldsymbol{\Theta}_j$-iterates are fairly "close to each other". However, if the $\boldsymbol{\Theta}_j$ drift apart, due to e.g. a bad initial guess $\boldsymbol{\Theta}_0$, a fixed stepsize sequence can produce very inaccurate or even nonsense solutions to the ODE-system.

In order to find the minimum of the objective function $\phi(\boldsymbol{\Theta})$, for each iteration we compute $\boldsymbol{\Theta}$, the numerical solution $\mathbf{y}_i(\boldsymbol{\Theta})$ of the ODE-system (2) and the corresponding Jacobians (sensitivity matrices) $\mathbf{S}_i = \partial \mathbf{y}_i / \partial \boldsymbol{\Theta}$, $i = 0, 1, \ldots, m$, at the measurement points $\mathbf{t} = (t_0, t_1, , \ldots, t_m)^T$. However, when solving an ODE-system with a discretization method that uses a stepsize sequence $[\mathbf{h}]$, we do not in general obtain integration points which are the same as the time measurement points, here called the interpolation points. We therefore have to interpolate (by dense output), in order to obtain numerical values $\mathbf{y}_i$ and $\mathbf{S}_i$ at the interpolation points $t_i$.

In order to compactify the description of the algorithms in this section we introduce the following compound array notations:

$$\tilde{\mathbf{y}} = (\tilde{y}_0, \ldots, \tilde{y}_m)^T \tag{16}$$

$$\mathbf{y}(\boldsymbol{\Theta})^T = (\mathbf{y}_0(\boldsymbol{\Theta})^T, \ldots, \mathbf{y}_m(\boldsymbol{\Theta})^T) \tag{17}$$

$$\mathbf{S}(\boldsymbol{\Theta})^T = (\mathbf{S}_0(\boldsymbol{\Theta})^T, \ldots, \mathbf{S}_m(\boldsymbol{\Theta})^T) \tag{18}$$

$$\mathbf{r}(\boldsymbol{\Theta})^T = (\mathbf{r}_0(\boldsymbol{\Theta})^T, \ldots, \mathbf{r}_m(\boldsymbol{\Theta})^T), \quad \text{where} \quad \mathbf{r}_i = \mathbf{y}_i(\Theta) - \tilde{y}_i \tag{19}$$

Let $M = (m+1)n$. Then the dimensions of $\tilde{\mathbf{y}}$, $\mathbf{y}$ and $\mathbf{r}$ are $M \times 1$ and $\mathbf{S}$ is $M \times s$. Also let $\mathbf{W}$ be the diagonal matrix composed of the diagonals in $\mathbf{W}_0, \mathbf{W}_1, \ldots \mathbf{W}_m$.

With this notation, a high-level description of the initial-value algorithm used in the *diffpar* toolbox is as follows:

**1.** Initiate (give $\Theta_0$ and tuning parameters, e.g. *trust*); let iteration index $j = 0$;

**2.** Get information about the linearized model at $\Theta_j$;

**2.1** compute a stepsize sequence [**h**] for $\Theta_j$;

**2.2** use [**h**] to solve the ODE-system and the variational equations; interpolate to get $\mathbf{y}_i$ and $\mathbf{S}_i$ at the measurement points $t_i$; compute **r**;

**2.3** solve the approximating linear least squares problem

$$\min_{\Delta\Theta} \frac{1}{2}(\mathbf{S}\Delta\Theta - \mathbf{r})^T\mathbf{W}(\mathbf{S}\Delta\Theta - \mathbf{r});$$

**2.4** if regularization is needed, let $trust = 1$ else $trust = 0$;

**2.5** check which perturbed problem is solved by $\Theta_j$; tabulate info about $\Theta_j$;

**while** $\Theta_j$ does not solve a slightly perturbed problem

**if** $trust = 0$ **then**

**3.** Compute $\Delta\Theta$ that gives descent along the Gauss-Newton direction with respect to a quadratic merit function;

**3.1** compute weights $\mathbf{W}_{\text{merit}}$ that makes $\Delta\Theta$ an acceptable descent direction of the merit function $\frac{1}{2}(\mathbf{y}(\Theta) - \tilde{\mathbf{y}})^T\mathbf{W}_{\text{merit}}(\mathbf{y}(\Theta) - \tilde{\mathbf{y}})$ (see section 3.2.4);

**3.2** compute a steplength $\alpha$ such that the real refinement is sufficiently much larger than the predicted refinement (see section 3.2.5); Let $\Delta\Theta_j = \alpha\Delta\Theta$;

**else**

**4.** Compute $\Delta\Theta$ that solves the subspace trust region problem;

**4.1** compute $\delta$, $\rho$ and $\Delta\Theta_j$ such that the real refinement is sufficiently much larger than the predicted refinement, i.e. solve (see section 3.2.7)

$$\min_{\Delta\Theta} \frac{1}{2}(\mathbf{S}\Delta\Theta - \mathbf{r})^T\mathbf{W}(\mathbf{S}\Delta\Theta - \mathbf{r})$$

$$\text{s.t.} \quad \|\mathbf{R}\Delta\Theta\| \leq \delta$$

where $\mathbf{R}\Delta\Theta = \alpha \cdot \text{diag}\,(d_1^{-1}, .., d_\rho^{-1}, 0, .., 0)\mathbf{Q}^T\mathbf{W}^{1/2}\mathbf{r}$ and $\mathbf{W}^{1/2}\mathbf{S} = \mathbf{Q}\begin{pmatrix}\mathbf{D}\\ \mathbf{0}\end{pmatrix}\mathbf{R}\mathbf{P}^T$

**end if;**

**5.** Let $\Theta_{j+1} = \Theta_j + \Delta\Theta_j \quad ; j = j + 1;$

**6.** Get information about the linearized model at $\Theta_j$;

**6.1** compute a new stepsize sequence [**h**] if necessary;

**6.2** The steps 2.2–2.5 are repeated with the current $j$-value;

**end while;**

### *3.2 Numerical solution of the minimization problem*

In order to find the minimum of the objective function (1), which is the square of a weighted norm of the residuals

$$\phi(\boldsymbol{\Theta}) = \frac{1}{2}\sum_{i=0}^{m} \mathbf{r}_i(\boldsymbol{\Theta})^T \mathbf{W}_i \mathbf{r}_i(\boldsymbol{\Theta}) = \frac{1}{2}\mathbf{r}(\boldsymbol{\Theta})^T \mathbf{W}\mathbf{r}(\boldsymbol{\Theta}) \tag{20}$$

where $\mathbf{r}_i(\boldsymbol{\Theta})$ and $\mathbf{r}(\boldsymbol{\Theta})$ are defined in (19), a modified Gauss-Newton algorithm is used. In each iteration, performed with this method, the following linear least squares problem is solved:

$$\min_{\Delta\boldsymbol{\Theta}} \frac{1}{2}(\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r})^T \mathbf{W}(\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r}) \tag{21}$$

which is constrained or unconstrained depending on whether regularization is used or not (see section 3.2.7).

*3.2.1 Why we choose the Gauss-Newton method.* For nonzero residual problems the Gauss-Newton method converges only linearly, while quasi-Newton methods have superlinear convergence rate. We still think that a version of the Gauss-Newton method is preferable for least squares problems in ODE's for the following reasons:

(i) A quasi-Newton method gets superlinear convergence rate because it builds up information about second derivatives from the first order information known to us. Hence it tries to construct a matrix $\mathbf{B}$ that approximates the true Hessian $\mathbf{S}^T\mathbf{S} + \sum_{i,q} r_i^{(q)} \nabla^2_{\Theta} y_i^{(q)}$. However, this turns out to be a formidable task when estimating parameters in ODE-systems since the second term in the Hessian varies much from one iteration to another. Besides, the exact Hessian is often indefinite until $\boldsymbol{\Theta}_j$ is fairly close to the solution, see Ramsin and Wedin [25].

(ii) The Gauss-Newton method has quadratic convergence only for zero residuals. However, far away from the solution $\Theta^*$= argmin $\phi(\Theta)$, the convergence behavior for problems with zero residuals is similar to problems with nonzero residuals, since the quadratic terms dominate there. Only close to $\Theta^*$ the convergence rate is linear.

(iii) The convergence rate of the Gauss-Newton method without regularization is a differential geometrical quantity that only depends on the curvature of the surface $\mathbf{y}(\boldsymbol{\Theta})$ with respect to the residual $\mathbf{r}^*$ at the solution $\Theta^*$, see Lindström and Wedin [21]. This fact has several useful consequences:

(iiia) We get exactly the same asymptotic convergence whether we use a logarithmic scale or not. This is extremely useful since choosing a logarithmic scale for $\Theta_i$ means adding the condition $\Theta_i > 0$, implicitly. Very often, such as in chemical kinetics, the parameter $\Theta_i$ corresponds to a rate that has to be positive. In this simple way we can bound the convergence area to positive $\Theta_i$-values. Note that this approach is superior to a method that uses a simple active set strategy for $\Theta_i \geq 0$ since such a strategy might take us to the boundary with $\Theta_i = 0$ and stay there if there is a local minimum of the constrained problem with $\Theta_i = 0$. In a way this simple device has turned out to be a very powerful device in the program. Hence, we get exactly the same asymptotic convergence rate whether we use a logarithmic scale or not.

(iiib) In the Gauss-Newton method, the iterates are repelled from stationary points that are not local minima — a good property since stationary points that are not local minima are common for nonlinear least squares problems and we don't want to end up at such a point.

(iiic) A least squares problem with $\tilde{\mathbf{y}}$ changed to $\tilde{\mathbf{y}} + \tau(\mathbf{y}^* - \tilde{\mathbf{y}})$ still has a stationary point at $\boldsymbol{\Theta}^*$. We get zero residuals for $\tau = 1$ and another least squares problem with residuals as large as the given problem for $\tau = 2$. If the nonlinearity of the surface is so large that this second problem does not have a minimum at $\boldsymbol{\Theta}^*$, the Gauss-Newton method will generate a sequence $\boldsymbol{\Theta}_j$ that close to $\boldsymbol{\Theta}^*$ will oscillate around $\boldsymbol{\Theta}^*$ and the convergence should be speeded up in some way, possibly as described by Ruhe [27]. We have not yet included such a convergence acceleration in the current program system. It should be born in mind that if the problem is so nonlinear that the local minimum depends on the direction of the residual, we cannot use standard statistical methodology to analyze the solution $\boldsymbol{\Theta}^*$. In fact, it is most likely that we have ended up at a local minimum that is not the one we are looking for. We should then try a better start vector $\boldsymbol{\Theta}_0$.

*3.2.2 Updating the stepsize sequence.* To be able to assure global convergence of the $\boldsymbol{\Theta}_j$-sequence to a first order Kuhn-Tucker point, i.e. a point where the linear least squares problem (21) has the solution $\Delta\boldsymbol{\Theta} = 0$, the stepsize sequence [**h**] is only allowed to be changed a limited number of times. Otherwise a more sophisticated framework is needed for the optimization algorithm.

*3.2.3 On weights and the merit function.* There are no bounds on the weights allowed in the nonlinear least squares problem. It is even possible to use an infinite weight, i.e. assume that some components of $\tilde{\mathbf{y}}_i$ satisfies the differential equation exactly at $t_i$. Then this should be a useful device, but it puts some restriction on the algorithm. First we have to use a method for the linear least squares problem (21) that is stable with respect to rounding errors for infinite weights. This is achieved by sorting the equations so that the weights in $\mathbf{W}$ appear in descending order, $w_1 \geq w_2 \geq \cdots \geq w_M \geq 0$. Then use the modified

QR decomposition introduced by Gulliksson and Wedin [15]. Make a decomposition

$$\mathbf{S} = \mathbf{Q}_1\mathbf{Q}_2 \ldots \mathbf{Q}_M \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{P}^T \tag{22}$$

with $\mathbf{Q}_j\mathbf{W}^{-1}\mathbf{Q}_j = \mathbf{W}^{-1}$ and $\mathbf{Q}_j^2 = I$, insert into (21) and we get

$$\Delta\boldsymbol{\Theta} = \mathbf{P} \begin{pmatrix} \mathbf{R}^{-1} \\ \mathbf{0} \end{pmatrix} \mathbf{Q}_M \ldots \mathbf{Q}_1\mathbf{r} \tag{23}$$

Here $\mathbf{R}$ is an upper triangular matrix with decreasing diagonal elements in the 2-norm, and $\mathbf{P}$ is a perturbation matrix that is used to get a matrix $\mathbf{R}$ with the above mentioned property.

If there are infinite weights there is no objective function whose decrease can be studied. Even if there are no infinite weights but, as is more natural, some very large weights in the objective function (20), it makes a very poor merit function. When the weight $w_i$ is large it is natural to choose the starting point $\boldsymbol{\Theta}_0$ such that the $i$th component of $\mathbf{y}$ and $\tilde{\mathbf{y}}$ are almost equal. This choice will often lead to unnecessarily small steps $\alpha\Delta\boldsymbol{\Theta}$ in the optimization process just to assure that the objective function decreases. The difficulties that we meet are similar to those that occur when we use a penalty function with very large weights for a constrained optimization problem.

Hence it is natural to use smaller weights in the merit function than those in the objective function. We can use the framework that was developed in Lindström and Wedin [21] and Gulliksson, Söderkvist and Wedin [14] for constrained least squares problems to compute the weights in a diagonal weight matrix $\mathbf{W}_{\text{merit}}$ that will assure convergence.

*3.2.4 Conditions on the weights in the merit function.* We require that $\Delta\boldsymbol{\Theta}$ is a good descent direction of (21) with $\mathbf{W}$ changed to $\mathbf{W}_{\text{merit}}$. This corresponds to the requirement that

$$\min_{\alpha} \frac{1}{2}(\alpha\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r})^T\mathbf{W}_{\text{merit}}(\alpha\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r}) \tag{24}$$

has a solution $\alpha$ that is close to 1; our requirement is $\alpha \geq 0.8$, see Lindström and Wedin [21]. We also have some rather mild requirements on how small the elements of $\mathbf{W}_{\text{merit}}$ are allowed to become. The elements are also bounded from above by those of $\mathbf{W}$. The condition has the form

$$\mathbf{W} \geq \mathbf{W}_{\text{merit}} \geq \mathbf{W}_{\text{memo}}$$

where the diagonal matrix $\mathbf{W}_{\text{memo}}$ is updated in each step. Under these constraints we require that $\|\mathbf{W}_{\text{merit}}\|$ is as small as possible.

*3.2.5 The steplength $\alpha$.* The predicted refinement between iteration $j$ and $j+1$ equals:

$$\text{pred} = \frac{1}{2}\mathbf{r}^T\mathbf{W}_{\text{merit}}\mathbf{r} - \frac{1}{2}(\alpha\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r})^T\mathbf{W}_{\text{merit}}(\alpha\mathbf{S}\Delta\boldsymbol{\Theta} - \mathbf{r}) \tag{25}$$

which is compared to the real refinement

$$\text{ref} = \frac{1}{2}\mathbf{r}^T\mathbf{W}_{\text{merit}}\,\mathbf{r} - \frac{1}{2}\mathbf{r}(\boldsymbol{\Theta} + \alpha\Delta\boldsymbol{\Theta})^T\mathbf{W}_{\text{merit}}\,\mathbf{r}(\boldsymbol{\Theta} + \alpha\Delta\boldsymbol{\Theta}) \tag{26}$$

We try a sequence $1, 0.5, (0.5)^2, (0.5)^3, \ldots$ for $\alpha$ until $\text{ref} \geq \tau \cdot \text{pred}$, where $\tau$ is a tuning parameter we set to 0.1. The condition assures convergence over $\boldsymbol{\Theta}$ of the Gauss-Newton method with steplength $\alpha$ for $\Delta\boldsymbol{\Theta} = \mathbf{S}^+\mathbf{r}$, if the condition number $\kappa(\mathbf{S}) = \|\mathbf{S}\|\|\mathbf{S}^+\|$ is bounded.

*3.2.6 On regularization of the linearized problem.* When regularization is needed, the objective function (20) is used as merit function. In the discussion below we have eliminated weights by replacing $\mathbf{W}^{1/2}\mathbf{y}(\boldsymbol{\Theta})$ and $\mathbf{W}^{1/2}\tilde{\mathbf{y}}$ for $\mathbf{y}(\boldsymbol{\Theta})$ and $\tilde{\mathbf{y}}$.

In an early version of *diffpar*, we used an ordinary *trust region method* for problem (20). To avoid computing $(\mathbf{S}^T\mathbf{S} + \mu\mathbf{I})^{-1}\mathbf{S}^T\mathbf{r}$ over and over again, we now use an idea proposed by Osborne [24]. First we make a $QR$-decomposition of $\mathbf{S}$ with column permutations to get

$$\mathbf{S} = \mathbf{Q}\begin{pmatrix}\mathbf{D}\\ \mathbf{0}\end{pmatrix}\mathbf{R}\mathbf{P}^T \tag{27}$$

where $\mathbf{D}$ is a diagonal matrix with decreasing elements, $\mathbf{P}$ a permutation matrix and $\mathbf{R}$ an upper triangular matrix with unit elements in the diagonal and all nondiagonal elements smaller than 1. This structure of $\mathbf{R}$ implies that the condition number of $\mathbf{R}$ is bounded by a constant that only depends on $s$. Hence it is sufficient to stabilize the diagonal matrix $\mathbf{D} = \text{diag}\,(d_1, d_2, \ldots, d_s)$. This is much less expensive than to stabilize $\mathbf{DR}$. Let $\mathbf{b}$ be the first $s$ components of $\mathbf{Q}^T\mathbf{r}$ and let $\mathbf{x}$ be the solution of the regularized problem

$$\min_x \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2 \tag{28}$$

$$\text{s.t.} \quad \|\mathbf{x}\|_2 \leq \Delta$$

where $\Delta\boldsymbol{\Theta} = \mathbf{P}\mathbf{R}^{-1}\mathbf{x}$ is varied in the usual way until we reach a $\Delta\boldsymbol{\Theta}$ that makes the real refinement sufficiently large compared to the predicted refinement.

This method guarantees convergence of the Gauss-Newton method even if $\mathbf{S}$ becomes singular, but the convergence can be slow. That was exactly what happened for Bock's almost singular problem, see section 4.3. The convergence was slow compared to that reported by Bock [6], who use a truncated $QR$-method for regularization. In a method of their kind the smallest elements in the diagonal matrix $\mathbf{D}$ are ignored and the Gauss-Newton direction is approached with the direction

$$\mathbf{p} = \mathbf{P}\mathbf{R}^{-1}\ \text{diag}\,(1/d_1, 1/d_2, \ldots 1/d_\rho, 0, \ldots, 0)\mathbf{b} \tag{29}$$

A method of this kind, where the search direction $\mathbf{p}$ is confined to certain subspaces, is in Lindström and Wedin [21] called a *subspace minimization method.* A few truncation rules are derived from the natural convergence condition that the angle between the search

direction $\mathbf{p}$ and the negative gradient of the object function shall be bounded away from $\pi/2$. Then heuristics is used to find the suitable rank $\rho$.

We propose the following *subspace trust region approach* that varies $\|\mathbf{x}\|$ in (29) continuously but confines $\mathbf{x}$ to certain subspaces.

$$\begin{aligned} &\min_x \ \|\mathbf{D}x - \mathbf{b}\|_2 \\ &\text{s.t.} \ \ \|\mathbf{x}\|_2 \leq \Delta \\ &\text{and} \ \ \mathbf{x}^T = \alpha(b_1/d_1, \ldots b_\rho/d_\rho, 0 \ldots, 0), \quad \text{for} \quad \rho = s, s-1, \ldots, 1 \end{aligned} \tag{30}$$

If $\Delta$ is varied as usual this approach can be shown, see Wedin and Edsberg [31] to generate a globally convergent method. The computational cost is smaller for problem (30) than for problem (28), but the main advantage of problem (30) is that this problem for large gaps in the diagonal of $\mathbf{D}$ usually has the solution in an inner point which makes it possible to reduce $\|\mathbf{x}\|_2$ to exactly the right size.

For Bock's almost singular problem with sufficiently large nonzero residual there is a striking difference and we get convergence in 6–8 iterations depending on the termination criterion. The predicted refinement for subspace trust region method gives as good a predicted refinement as the ordinary trust region approach.

*3.2.7 Termination criterion.* Let $\Delta\boldsymbol{\Theta}$ be computed as the solution of the approximating linear least squares problem (21). Obviously the nonlinear least squares problem

$$\min_{\boldsymbol{\Theta}} \frac{1}{2}(\mathbf{y}(\boldsymbol{\Theta}) - (\tilde{\mathbf{y}} - \mathbf{S}\Delta\boldsymbol{\Theta}))^T \mathbf{W}(\mathbf{y}(\boldsymbol{\Theta}) - (\tilde{\mathbf{y}} - \mathbf{S}\Delta\boldsymbol{\Theta})) \tag{31}$$

has a stationary point that generally is a local minimum at the current iterate $\boldsymbol{\Theta}$. Problem (31) can be seen as a parameter estimation problem of the same kind as the given problem but with slightly perturbed data, $S\Delta\Theta$. It is natural to terminate the computations when its absolute norm $\|\mathbf{S}\Delta\boldsymbol{\Theta}\|$ or relative norm $\|\mathbf{S}\Delta\boldsymbol{\Theta}\|/\|y - \tilde{y}\|$ is smaller then a given tolerance *Tol*.

### *3.3 Numerical solution of the initial value problem*

In *diffpar*, the following two discretization methods are available:

1. a Runge-Kutta 3(2)-method for non-stiff problems
2. a third order BDF-method for stiff problems.

The ODE-system (2) is solved with one of these methods and the corresponding sensitivity matrix $S$, building up the Jacobian (18),

$$\mathbf{S} = \begin{pmatrix} \dfrac{\partial \mathbf{y}}{\partial \mathbf{k}} & \dfrac{\partial \mathbf{y}}{\partial \mathbf{y}_0} \end{pmatrix} \tag{32}$$

is obtained by solving the linearized variational equations:

$$\dot{\mathbf{S}} = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t, \mathbf{y}, \mathbf{k})\,\mathbf{S} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{k}}(t, \mathbf{y}, \mathbf{k}) \quad \mathbf{0}\right), \qquad \mathbf{S}(0) = (\mathbf{S}_0 \quad \mathbf{I}). \tag{33}$$

In case we optimize with respect to the logarithmated $\mathbf{k}$-values, the corresponding sensitivity matrix $S_{\ln}$ is obtained from $S_{\ln} = S \cdot \operatorname{diag}(k)$.

In the Gauss-Newton algorithm we need to distinguish between two ways of choosing the stepsize sequence [$\mathbf{h}$] for numerical solution of the ODE-system:

1. automatic stepsize control, where [$\mathbf{h}$] is selected by keeping a local error estimate below a certain tolerance given as input.
2. prescribed stepsize control, where [$\mathbf{h}$] is given as input.

It should be noted that when one performs a parameter estimation with *diffpar*, most of the execution time is taken up by solving the variational equations. Next in computer effort is spent on QR-decomposition of the Jacobian (18) in order to find a search direction and on third place we have the solution of the ODE-system for a given value of $\boldsymbol{\Theta}$.

*3.3.1 The Runge-Kutta solver.* The Runge-Kutta algorithm used in the toolbox is an explicit 3rd order, 3-stage method which uses a 2nd order formula for local error estimation and automatic/prescribed stepsize control. This method is suited for non-stiff problems requiring moderate accuracy. The method is discussed in the framework of general explicit Runge-Kutta methods in e.g. Hairer, Nørsett and Wanner [16], but is presented here since different stepsize strategies are needed.

Taking one step from $(t_i, \mathbf{y}_i)$ to $(t_{i+1}, \mathbf{y}_{i+1})$ is described by the formulas:

$$\begin{aligned} \mathbf{y}_{i+1} &= \mathbf{y}_i + h_i(\mathbf{q}_1 + 4\mathbf{q}_3 + \mathbf{q}_2)/6 \\ t_{i+1} &= t_i + h_i \end{aligned} \tag{34}$$

where

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{f}(t_i, \mathbf{y}_i, \mathbf{k}) \\ \mathbf{q}_2 &= \mathbf{f}(t_i + h_i, \mathbf{y}_i + h_i\mathbf{q}_1, \mathbf{k}) \\ \mathbf{q}_3 &= \mathbf{f}(t_i + h_i/2, \mathbf{y}_i + h_i(\mathbf{q}_1 + \mathbf{q}_2)/4, \mathbf{k}) \end{aligned} \tag{35}$$

The estimator $\mathbf{y}_{i+1}^{\text{est}}$ used to estimate the local error is obtained from

$$\mathbf{y}_{i+1}^{\text{est}} = \mathbf{y}_i + h_i(\mathbf{q}_1 + \mathbf{q}_2)/2 \tag{36}$$

The two formulas for $\mathbf{y}_{i+1}$ and $\mathbf{y}_{i+1}^{\text{est}}$ constitute an embedded Runge-Kutta formula.

*3.3.2 Automatic stepsize control for Runge-Kutta.* When automatic stepsize control is used, the stepsize selection is based on a local mixed absolute/relative error estimate $err$ and a given initial stepsize $h_0$. The norm $\|.\|$ used in this section is the maximum norm.

$$err = \|\mathbf{y}_{i+1}^{\text{est}} - \mathbf{y}_{i+1}\| / \max(\|\mathbf{y}_{i+1}\|, 1) \tag{37}$$

It can be shown that

$$\|\mathbf{y}_{i+1}^{\text{est}} - \mathbf{y}_{i+1}\| = Ch_i^3. \tag{38}$$

Hence

$$err \cdot \max(\|\mathbf{y}_{i+1}\|, 1) = Ch_i^3. \tag{39}$$

The new stepsize $h_{i+1}$ should fulfil

$$tol \cdot \max(\|\mathbf{y}_{i+1}\|, 1) = Ch_{i+1}^3 \tag{40}$$

where $tol$ is the user given local relative error tolerance. We obtain from (39) and (40)

$$h_{i+1} = h_i(tol/err)^{1/3}. \tag{41}$$

A safety factor 0.9 is introduced and we also do not want $h$ to increase too fast:

$$h_{i+1} = \min(h_{\max}, 0.9h_i(tol/err)^{1/3}) \tag{42}$$

which is the stepsize selection rule used in the Runge-Kutta method using automatic stepsize control in *diffpar*. When the stepsize sequence is given, i.e. prescribed stepsize control is used, the numerical solution of the ODE-system is obtained simply by using the formulas (34) and (35). The local error estimate, used to check the accuracy in the solution, is obtained from (36) and (37).

*3.3.3 Numerical solution of the variational equation.* In order to compute $\mathbf{S}(\boldsymbol{\Theta})$, the variational equations (33) in matrix form are solved using the same stepsize sequence as for the corresponding ODE-system:

$$\mathbf{S}_{i+1} = \mathbf{S}_i + h_i(\mathbf{Q}_1 + 4\mathbf{Q}_3 + \mathbf{Q}_2)/6 \tag{43}$$

where $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ are matrices computed simularly as $\mathbf{q}_1$, $\mathbf{q}_2$ and $\mathbf{q}_3$ in (35). The ODE-system and the variational equations are solved sequentially in each step, i.e. suppose $t_i$, $\mathbf{y}_i$ and $\mathbf{S}_i$ are given. We then compute $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ and $\mathbf{y}_{i+1}$ from (3.17) and (3.18) and proceed with $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ and $\mathbf{S}_{i+1}$. It is easy to prove that this computational scheme will generate a Jacobian $\mathbf{S}(\boldsymbol{\Theta})$, to be used in the Gauss-Newton algorithm, that is consistent with $\mathbf{y}(\boldsymbol{\Theta})$, i.e.

$$\frac{\partial \mathbf{y}}{\partial \boldsymbol{\Theta}} = \mathbf{S}(\boldsymbol{\Theta}). \tag{44}$$

In order to interpolate the solution $\mathbf{y}(\Theta)$ and the Jacobian $\mathbf{S}(\Theta)$, at the interpolation points a dense output scheme according to Hairer, Nørsett and Wanner [16] is used. At an interpolation point $t_i + \alpha h_i$ in the interval $(t_i, t_{i+1})$, the following Newton interpolation formula is used:

$$\mathbf{y}(t_i + \alpha h_i) = \mathbf{y}_i + \alpha h_i(\mathbf{f}_i + \alpha(\mathbf{d} - \mathbf{f}_i) + \alpha(\alpha - 1)(\mathbf{f}_{i+1} - 2\mathbf{d} + \mathbf{f}_i)) \tag{45}$$

where

$$\mathbf{d} = (\mathbf{y}_{i+1} - \mathbf{y}_i)/h_i. \tag{46}$$

Based on this formula we obtain 3rd order interpolation of $\mathbf{y}(\Theta)$ and $\mathbf{S}(\Theta)$.

*3.3.4 The BDF-solver.* The Backwards Differentiation Formula (BDF) method used in the toolbox is a 3rd order, 3-step method using automatic or prescribed stepsize control. This method is suited for stiff problems requiring moderate accuracy. The method is described in the general framework of BDF-methods in e.g. Hairer, Nørsett and Wanner [17].

The stepsize strategy in this implementation is arcwise constant, i.e. 3 consecutive steps of equal size are taken in an arc. This kind of implementation reduces the computations involved in the change of stepsize, and is based on an idea by Dahlquist [9]. The method belongs to the class of linear multistep methods and can be written as:

$$\nabla \mathbf{y}_{i+1} + \frac{1}{2}\nabla^2 \mathbf{y}_{i+1} + \frac{1}{3}\nabla^3 \mathbf{y}_{i+1} = h_i \mathbf{f}(t_{i+1}, \mathbf{y}_{i+1}, \mathbf{k}) \tag{47}$$

$$t_{i+1} = t_i + h_i. \tag{48}$$

An equivalent formulation of (47) is the multivalue representation

$$\nabla^3 \mathbf{y}_{i+1} = \frac{6h_i}{11}\left[\mathbf{f}(t_{i+1}, \nabla^3 \mathbf{y}_{i+1} + \nabla^2 \mathbf{y}_i + \nabla \mathbf{y}_i + \mathbf{y}_i) - \frac{1}{h_i}\left(\frac{3}{2}\nabla^2 \mathbf{y}_i + \nabla \mathbf{y}_i\right)\right] \tag{49}$$

and its corresponding Nordsieck matrix of differences:

$$\mathbf{y}_i^{\text{dif}} = [\nabla^3 \mathbf{y}_i, \nabla^2 \mathbf{y}_i, \nabla \mathbf{y}_i, \mathbf{y}_i]. \tag{50}$$

After solving (49) with respect to $\nabla^3 \mathbf{y}_{i+1}$, replacing the first column of $\mathbf{y}_i^{\text{dif}}$ by $\nabla^3 \mathbf{y}_{i+1}$ and then making a columnwise cumulative summation of $\mathbf{y}_i^{\text{dif}}$, $\mathbf{y}_{i+1}^{\text{dif}}$ is obtained, where $\mathbf{y}_{i+1}$ is available in the last column.

From the Nordsieck matrix there is an easy way of constructing the vector polynomial $\mathbf{p}(t)$, interpolating the numerical solution at a point $t = t_i - \nu h$ in an arc, generated with stepsize $h$ and consisting of the points $(t_{i-k}, \mathbf{y}_{i-k})$, $k = 0, 1, 2, 3$ as follows:

$$\mathbf{p}(t_i - \nu h) = \mathbf{y}_i^{\text{dif}}\, \mathbf{q}(\nu) \tag{51}$$

where

$$\nu = (t_i - t)/h \qquad (0 \leq \nu \leq 3) \tag{52}$$

$$\mathbf{q}(\nu) = (-\nu(\nu-1)(\nu-2)/6, \nu(\nu-1)/2, -\nu, 1)^T \tag{53}$$

The local error estimate $err_i$, which is computed after each arc is based on the formula:

$$err_i = \frac{1}{4} \| \nabla^4 \mathbf{y}_i \| \tag{54}$$

*3.3.5 Automatic stepsize control for BDF.* In addition to $\mathbf{y}_0$, two more vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ are needed to start the difference formula (49). These values are obtained by computing the first arc with the Runge-Kutta-method described in 3.3.1. Based on a given stepsize $h_0$ and tolerance $tol$, the first arc is computed from the formula (34) and (35).

When the first arc has been completed, the BDF-formula (49) is used to generate the rest of the $\mathbf{y}_i(\Theta)$-values, $i = 4, 5, \ldots$, with arcwise constant steps.

The automatic stepsize selection, which is performed after each arc, is based on the following decisions concerning the estimated local error $err$ and the given tolerance $tol$:

1. accept (if $err \leq tol$) or reject (if $err > tol$) the previous arc
2. if the arc is rejected, decide which of the stepsizes $h/2^i$, $i = 1, 2, 3$ should be used in the recomputation of the previous arc
3. if the arc is accepted, decide which of the stepsizes, $2h$, $h$, or $h/2^i$, $i = 1, 2, 3$ should be used in the computation of the next arc.

If the stepsize is changed in a point $t_i$ at the end of an arc, $\mathbf{y}_i^{\text{dif}}$ must be updated to differences based on $\mathbf{y}$-values corresponding to the new step size. Hence, if the stepsize is doubled, we need two $\mathbf{y}$-values corresponding to the new step size before the arc. Denote by $\mathbf{y}_i^{2h} = [\mathbf{y}_{i-3}, \mathbf{y}_{i-2}, \mathbf{y}_{i-1}, \mathbf{y}_i]$ the four consecutive $\mathbf{y}$-values corresponding to the new stepsize. Then transform to the difference matrix:

$$\mathbf{y}_i^{\text{dif}} = \mathbf{y}_i^{2h} \begin{pmatrix} -1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ -3 & -2 & -1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{55}$$

and then proceed with (49). If the stepsize is halved, the interpolation polynomial (51) is used to compute two $\mathbf{y}$-values, corresponding to the new stepsize, in the arc. These values are obtained by putting $\nu = 3/2, 1, 1/2, 0$ in (53):

$$\mathbf{y}_i^{h/2} = \mathbf{y}_i^{\text{dif}} \begin{pmatrix} 1/16 & 0 & -1/16 & 0 \\ 3/8 & 0 & -1/8 & 0 \\ -3/2 & -1 & -1/2 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{56}$$

The new Nordsieck matrix, corresponding to the halved stepsize, is then obtained from the formula (50).

When the new stepsize $h_i$ has been decided, the arc is computed by solving the multivalue representation of the multistep formula (49) with a Newton-method using the iteration matrix $\mathbf{I} - (6h_i/11)\partial\mathbf{f}/\partial\mathbf{y}$, where the Jacobian $\partial\mathbf{f}/\partial\mathbf{y}$ is evaluated at the first arcpoint in the arc. At the last step of an arc, two Newton-iterations are performed, at the first steps only one iteration is done. When the stepsize sequence is given, i.e. prescribed stepsize control is used, the numerical solution of the ODE-system is obtained by using the formula (49). The local error is estimated from (54).

*3.3.6 Numerical solution of the variational equations.* The variational equations (33) for a stiff system of ODE's are solved in a similar way as described in 3.3.4, after having formulated them in vector form. Denote by $\mathbf{s}_i$ the $i$th column of $\mathbf{S}$. The variational equations can then be written in compound vector form

$$\dot{\mathbf{s}} = \mathbf{F}_\mathbf{y}\mathbf{s} + \mathbf{f}_\Theta, \quad \mathbf{s}(0) = \mathbf{s}_0 \tag{57}$$

where

$$\mathbf{s}^T = \left(\mathbf{s}_1^T, \mathbf{s}_2^T, \ldots, \mathbf{s}_{n+p}^T\right) \tag{58}$$

$$\mathbf{F}_\mathbf{y} = \operatorname{diag}\left(\left\{\frac{\partial\mathbf{f}}{\partial\mathbf{y}}\right\}\right) \tag{59}$$

$$\mathbf{f}_\Theta^T = \left(\left(\frac{\partial\mathbf{f}}{\partial\Theta_1}\right)^T, \left(\frac{\partial\mathbf{f}}{\partial\Theta_2}\right)^T, \ldots, \left(\frac{\partial\mathbf{f}}{\partial\Theta_{n+p}}\right)^T\right). \tag{60}$$

In this vector-representation of the elements in the sensitivity matrix, $\mathbf{S}$ can be treated in multivalue form, just as the ODE-system itself, and the MATLAB-codes for the ODE-system and the variational equations are similar. Note, however, that (57) is linear in $\mathbf{s}$, so only one Newton iteration is necessary for all steps in an arc.

In order to interpolate the solution $\mathbf{y}(\Theta)$ and $\mathbf{S}(\Theta)$ at the interpolation points, dense output, based on the interpolation formula (51), is used.

## 4 TEST PROBLEMS AND RESULTS

In this section five test problems are presented. The first three examples, which all have the property that the parameters enter linearly in the right hand side, are taken from earlier references while the last two, coming from industrial applications, are real life problems, nonlinear in all respects.

Only in the first example data are given. In the other examples, data are generated by solving the ODE-system for a given value of the parametervector $\Theta$ giving a compound solution vector $y(\Theta)$. Perturbations, which simulate measurement errors, are added to this vector $y(\Theta)$.

The perturbations are computed in such a way that normal distributed noise with mean value zero and standard deviation one is generated by Matlab's pseudo-random number generator and then multiplied by a scalar $\delta$ that can be varied. The resulting vector is then projected onto the space orthogonal to the range of the Jacobian $S$ evaluated at the given parameter vector $\Theta$ and then finally added to $y(\Theta)$. In this way we obtain a residuum $\phi(\Theta) > 0$ but the estimate $\Theta_{\text{est}}$ fulfils $\Theta_{\text{est}} = \Theta$. The tolerance *Tol* (see 3.2.7) used in all experiments is $10^{-3}$. The weights are all chosen to one, except for one testcase in the first example below.

### 4.1 *Barnes' problem*

This problem is mentioned in the introduction and consists of 2 ODE's with 3 parameters in the right hand sides and another 2 parameters if the initial values are not known, see (3) and (4). This ODE-system and the given data have been used by e.g. Tjoa and Biegler [29], Domselaar and Hemker [10] and Varah [30] as a test problem for parameter estimation in ODE-systems. This is a small non-stiff problem but has its interest in the fact that there are several "solutions" corresponding to different local minima of the objective function. The starting values of the parameters decide which local minimum will be hit, and the corresponding residual is fairly large. Even the "best" solution, i.e. the parameter vector giving the global minimum, has a rather large residual.

For this example the regularization facility in the Gauss-Newton algorithm is seldom used; the full Gauss-Newton step, corresponding to the solution of (21) can be used. Only in the case of large weights regularization according to the algorithm given in 3.1 is necessary.

With the starting value $\Theta_0 = (1, 1, 1)^T$, the testcases a) and d) converge in 7 and 5 iterations respectively. With the same starting value and the weights ${}_{10}6$ given to the last datapoint in testcase c) we have convergence in 21 iterations. Testcase b), finally, converges in 6 iterations from the starting value $\Theta_0 = (1, 1, 1, 1, 0.3)^T$, where the last two components are the measured initial values $a$ and $b$.

### 4.2 *Robertson's problem*

Robertson's problem is a classical example of a stiff ODE-system. It consists of 3 ODE's and 3 parameters according to:

$$\frac{dy_1}{dt} = -k_1 y_1 + k_2 y_2 y_3$$

$$\frac{dy_2}{dt} = k_1 y_1 - k_2 y_2 y_3 - k_3 y_2^2$$

$$\frac{dy_3}{dt} = k_3 y_2^2$$

The initial condition is $y(0) = (1, 0, 0)^T$ and the time interval is $(0, 1000)$. From the parametervector $\Theta$ with components $k_1 = 0.04$, $k_2 = 10^4$, $k_3 = 3.10^7$, data are generated

by solving the ODE-system and adding perturbations of size $\delta = 0.01$, to the solution at the points $t = 0{:}0.001{:}0.01, 0.02{:}0.01{:}0.1, 0.2{:}0.1{:}1, 10{:}100{:}1000$.

The problem is of interest since it shows the necessity of iterating in $\ln(\Theta)$, since negative components in $\Theta$ make the ODE-solution unstable. Iteration in $\ln(\Theta)$ converges from the starting value $\Theta_0 = (1, 1, 1)^T$ to an estimate. Even with zero-residual ($\delta = 0$), however, it is difficult to estimate all three parameters accurately. After 12 iterations we obtain $\Theta_{\text{est}} = (3.99_{10} - 2, 3.07_{10}3, 2.85_{10}6)^T$, i.e., $k_1$ is estimated accurately, $k_2$ and $k_3$ have large errors. This is in agreement with the results in Curtis and Edsberg [8], where it is demonstrated that nonlinear combinations, like $k_3/k_2^2$, are well-determined but not the parameters themselves. Similar results are obtained with $\delta = 0.01$ perturbed data.

### 4.3 *Bock's problem*

This problem was introduced by Bock [6] and consists of 7 non-stiff ODE's with 11 unknown parameters in the right hand sides of the ODE's. The problem comes from a chemical application: denitrogenation of pyridine.

$$\frac{dy_1}{dt} = -k_1 y_1 + k_9 y_2$$

$$\frac{dy_2}{dt} = k_1 y_1 - k_2 y_2 - k_3 y_2 y_3 + k_7 y_4 - k_9 y_2 + k_{10} y_4 y_6$$

$$\frac{dy_3}{dt} = k_2 y_2 - k_3 y_2 y_3 - 2k_4 y_3^2 - k_6 y_3 + k_8 y_5 + k_{10} y_4 y_6 + 2k_{11} y_5 y_6$$

$$\frac{dy_4}{dt} = k_3 y_2 y_3 - k_5 y_4 - k_7 y_4 - k_{10} y_4 y_6$$

$$\frac{dy_5}{dt} = k_4 y_3^2 + k_5 y_4 - k_8 y_5 - k_{11} y_5 y_6$$

$$\frac{dy_6}{dt} = k_3 y_2 y_3 + k_4 y_3^2 + k_6 y_3 - k_{10} y_4 y_6 - k_{11} y_5 y_6$$

$$\frac{dy_7}{dt} = k_6 y_3 + k_7 y_4 + k_8 y_5$$

The initial vector is $y(0) = (1, 0, 0, 0, 0, 0, 0)^T$ and the time interval is $(0, 2.5)$. The parameter vector consists of rate constants with the values $k_1 = 1.81$, $k_2 = 0.894$, $k_3 = 29.4$, $k_4 = 9.21$, $k_5 = 0.058$, $k_6 = 2.43$, $k_7 = 0.0644$, $k_8 = 5.55$, $k_9 = 0.0201$, $k_{10} = 0.577$, $k_{11} = 2.15$. When iterating in $\Theta$, *diffpar* converges to an estimate having some rate constants negative. On the short time interval given, the instability in the ODE-solution due to these components, will not cause problems for the least squares method.

With the starting value $\Theta_0 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T$ and perturbed data ($\delta = 0.01$) we get convergence in 9 iterations when iterating in $\ln(\Theta)$. Two of the rate constants $k_4$ and $k_{11}$ are hard to determine accurately and regularization with the subspace trust region method is necessary.

With the usual trust region method, however, the convergence is very slow close to the minimum, since the Gauss-Newton step must be reduced very much before the predicted refinement is sufficiently much smaller than the real refinement.

### 4.4 *The bioprocess problem*

This problem has its origin in a real life problem modeling the control of a recombinant microbial process coming from Kabi Peptide Hormones RD, Pharmacia AB, Sweden. Parameter estimation for the model linearized around a trajectory is studied in a stochastic framework in Axelsson [2]. The full nonlinear problem is recently treated by Mari [22].

The system of 3 ODE's in this problem turns out to be stiff. There are 7 parameters in the right hand side of the ODE's, but only 4 of them are to be estimated, the remaining 3 are considered as known.

$$\frac{dy_1}{dt} = -\frac{k_1 y_1}{y_1 + k_2} e^{y_3} + k_3 u(t)$$

$$\frac{dy_2}{dt} = k_4 \left( \frac{k_1 y_1}{y_1 + k_2} - \frac{k_5 k_6}{y_2 + k_6} \right) e^{y_3} - y_2 u(t)$$

$$\frac{dy_3}{dt} = \frac{k_7 y_1}{y_1 + k_2} \frac{k_6}{y_2 + k_6} - u(t)$$

The function $u(t)$ is a given control function, approximated by an interpolation polynomial of degree 9

$$u(t) = c_{10} t^9 + c_9 t^8 + \ldots c_2 t + c_1$$

where the coefficient vector $c$ is

$$c = (1.30_{10} - 3, 4.59_{10} - 4, -4.10_{10} - 4, 2.27_{10} - 4, -5.66_{10} - 5, ..$$

$$7.47_{10} - 6, -5.57_{10} - 7, 2.36_{10} - 8, -5.29_{10} - 10, 4.88_{10} - 12)$$

The initial value vector is $y(0) = (0.104, 0, 0.69)^T$. Typical estimated parameter values are $\Theta = (2, 0.70, 500, 0.39, 0.26, 10, 1.1)^T$. The state variable vector $y$ represents concentrations and is therefore non-negative. In case a concentration reaches zero level and remains zero some time the sensitivity matrix $S$ passes through a discontinuity and the Gauss-Newton method will not work without modification. We overcame this by testing *diffpar* on the time-interval (0, 15) where all components were strictly positive.

With the starting value $\Theta_0 = (2, 0.70, 500, 0.5, 0.25, 8, 1)^T$ (the first 3 parameters are known and held constant during the iterations) *diffpar* converges in 5 iterations to $\Theta$ given above.

### 4.5 *The shock-front problem*

This problem deals with the description of a broadly curved shockfront geometry, occurring in the modeling of the reaction zone in detonating explosives. The problem is presented in a report by Lee [19] and is communicated to us from Nitro-Nobel AB, Gyttorp, Sweden. It shows a lack of information problem occurring when the whole state vector cannot be measured.

The ODE-system consists of 2 nonstiff ODE's defined on the interval $(0, R)$:

$$\frac{d\phi}{dr} + \frac{\tan\phi}{r} = \frac{\kappa(D_0 \cos\phi)}{\cos\phi}, \quad \phi(0) = 0$$

$$\frac{dz}{dr} = -\tan\phi, \quad z(0) = z_0$$

where the curvature function $\kappa(D_n)$, $D_n = D_0 \cos\phi$ in the right hand side is computed from the equation (by Newton's method)

$$\frac{D_n}{D_{CJ}} = 1 - (\alpha_1 \kappa^{1/\nu} + \alpha_2 \kappa e^{\psi(D_{CJ} - D_n)})$$

If $D_0$ (the detonation velocity), $D_{CJ}$ (the Chapman-Jouquet velocity) and $z_0$ are known, there are 4 parameters $\Theta = (\alpha_1, \nu, \alpha_2, \psi)$ to estimate in $\kappa(D_n)$ from measurements of the $z$-component, the shockfront shape.

For only one $D_0$-value, the information to determine the $\kappa(D_n)$-curve is too limited; measurements from several shockfronts with different $D_0$- and $z_0$-values are needed to estimate the parameters. In our experiment we used three shock-fronts with $D_0 = 6.156, 5.835, 4.044$, the corresponding $z_0$-values $z_0 = 17.5, 8.4, 3.3$ and the common parameter value $D_{CJ} = 6.788$.

Typical estimated parameter values are $\Theta = (1.25, 1.76, 2.31, 0.41)^T$ obtained after 19 iterations with *diffpar* from the starting value $\Theta_0 = (2, 1.45, 1, 2)^T$.

## 5 CONCLUDING REMARKS

Some numerical tools, appropriate for parameter estimation in systems of ordinary differential equations, and implemented as a Matlab toolbox called *diffpar*, have been presented. A new method, the subspace trust region method for regularization of a nonlinear least squares problem, turns out to be a useful tool. The outcome of different tests on problems taken from earlier references and real life applications are discussed. The program and the test problems are available from the first author of the article. Future plans include generalization to differential-algebraic system and also incorporating a method for computation of a reasonable starting-value of the parameter vector.

REFERENCES


1. Aiken, R. (1985). *Stiff computation*. Oxford University Press.
2. Axelsson, J.P. (1988). Experimental techniques and data analysis to determine baker's yeast ethanol dynamics. *Anal. Chim. Acta*, **213**, 151–163.
3. Baden, N. and Villadsen, J. (1982). A family of collocation based methods for parameter estimation in differential equations. *Chemical Engineering Journal*, **23**, 1–13.
4. Bard, Y. (1974). *Nonlinear parameter estimation*. Academic Press.
5. Biegler, L., Damiano, J.J. and Blau, G.E. (1986). Nonlinear parameter estimation: A case study comparison. *American Institute of Chemical Engineering Journal*, **9**, 257–267.
6. Bock, H.G. (1983). Recent advances in parameter identification techniques for ODE's, *Numerical treatment of inverse problems in differential and integral equations*, 95–121, Birkhuser, 1983. Edited by Deuflhard, P. and Hairer, E.
7. Chance, E.M., Curtis, A.R., Jones, I.P. and Kirby, C.R. (1977). *FACSIMILE: a computer program for flow and chemistry simulation, and general initial value problem*. Report AERE-R 8775, AERE Harwell.
8. Curtis, A.R. and Edsberg, L. (1973). *Some investigations into data requirements for rate constant estimation*. Report AERE T.P. 554, AERE Harwell.
9. Dahlquist, G. Private communication.
10. Domselaar, van B. and Hemker, P. (1975). *Nonlinear parameter estimation in initial value problems*. Report NW 18/75, Mathematisch Centrum, Amsterdam.
11. Edsberg, L. and Wedin, P-Å. (1993). *Diffpar, a toolbox for parameter estimation in ODE-systems*. Report TRITA–NA–9308, Dep of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.
12. Ekstam, L. and Smed, T. (1987). *Parameter estimation in dynamic systems with application to power engineering*. Report, UPTEC 87 47 R, Institute of technology, Uppsala University.
13. Gear, C.W. (1983). Smooth numerical solutions of ordinary differential equations. *Numerical treatment of inverse problems in differential and integral equations*, 95–121, Birkhäuser, 1983. Edited by Deuflhard, P. and Hairer, E.
14. Gulliksson, M., Söderkvist, I. and Wedin, P-Å. (1993). Weighted nonlinear least squares with equality constraints. Accepted for publication in *SIAM J. Optimization*.
15. Gulliksson, M. and Wedin, P-Å. (1992). Modifying the QR-decomposition to constrained and weighted linear least squares. *SIAM J. Matrix Anal. Appl.*, **13**, 1298–1313.
16. Hairer, E., Norsett, S.P. and Wanner, G. (1993). *Solving ordinary differential equations I*. Springer.
17. Hairer, E., Norsett, S.P. and Wanner, G. (1991). *Solving ordinary differential equations II*. Springer.
18. Hemker, P.W. and Kok, J. (1993). *A project on parameter identification in reaction kinetics*. Report NM-R9301, CWI, Amsterdam.
19. Lee, J. (1990). *Detonation shock dynamics of composite energetic materials*. Thesis, Department of Materials and Metallurgical Engineering, New Mexico Institute of Mining and Technology, Socorro, New Mexico.
20. Leis, J. and Kramer, M. (1988). The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations. *Transaction of Mathematical Software*, **14**, 45–60.
21. Lindström, P. and Wedin, P-Å. (1988). *Methods and software for nonlinear least squares problems*. Report UMINF–133.88, Institute of information processing, University of Ume.
22. Mari, J. (1994). *Mathematical methods applied to biotechnical processes*. Report TRITA/MAT-94–43, Optimization and Systems Theory, KTH.
23. Nowak, U. and Deuflhard, P. (1985). Numerical identification of selected rate constants in large chemical reaction systems. *Applied Numerical Mathematics* **1**, 59–75.
24. Osborne, M. Private communication.
25. Ramsin, H. and Wedin, P-Å. (1977). A comparison of some algorithms for the nonlinear least squares problem. *BIT*, **17**, 72–90.
26. Rosenbrock, H.H. and Storey, C. (1966). *Computational techniques for chemical engineers*. Chapter 8. Pergamon Press, New York.
27. Ruhe, A. (1979). Accelerated Gauss–Newton algorithms for nonlinear least squares problems. *BIT*, **19**, 356–367.

28. Schittkowski, K. (1992). *NLSFIT: A FORTRAN code for parameter estimation in differential equations and explicit model functions*. Preprint, University of Bayreuth.
29. Tjoa, I., and Biegler, L. (1991). Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial Engineering Chemistry Research*, **30**, 376–385.
30. Varah, J.M. (1982). A spline least squares method for numerical parameter estimation in differential equations. *SIAM, Journal of Scientific and Statistical Computation*, **3**, 28–46.
31. Wedin, P-Å. and Edsberg, L. (1994). *A trust region method for subspace minimization*. Report TRITA–NA–9412, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.
32. Wikström, G. (1995). *On the computation of parameters that occur linearly in an ODE–model*. Report, Institute of information processing, University of Ume.
33. Williams, J. (1988). *Approximation and parameter estimation in ordinary differential equations*. Numerical analysis report no 171, Dep. of Mathematics, University of Manchester.