96 Mathcad[1] Worksheets for Data Fitting

Address:	Prof. Dr. K. Schittkowski Department of Mathematics University of Bayreuth D - 95440 Bayreuth
Phone:	+921 553278 (office) +921 32887 (home)
Fax:	+921 35557
E-mail:	klaus.schittkowski@uni-bayreuth.de
Web:	http://www.klaus-schittkowski.de
Date:	April 17, 2004

Abstract

The purpose of the paper is to introduce a set of data fitting test examples in form of Mathcad[2] worksheets. The availability of data fitting test problems is an important assumption to develop and test least squares codes, to learn how optimization routines behave, or to become familiar with implementation and user interface. The problems are taken from a collection of test examples for data fitting in dynamical systems, see Schittkowski [40]. The report contains a brief introduction and review on least squares methods, moreover a summary of 96 data fitting test problems that have been transferred to Mathcad and a detailed example. All worksheets can be downloaded from the home page of the author[3]. A particular advantage of executing these problems from Mathcad is the possibility to plot corresponding residuals easily.

[1] ©2003 Mathsoft Engineering & Education Inc.
[2] http://www.mathcad.com
[3] http://www.klaus-schittkowski.de

1 Introduction

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

We consider parameter estimation problems, where a model function $h(p,t)$ is available in explicit form, the so-called *fitting criterion*, depending on the parameter vector $p \in \mathbb{R}^{n_p}$ to be estimated and an additional variable t called *time*. We proceed from n_d experimental data given in the form

$$(t_i, y_i, w_i) \ , \ i = 1, \ldots, n_d \ \ , \tag{1}$$

where n_d time and corresponding measurement values are given. Moreover, we assume that there are suitable weighting factors $w_i \geq 0$ given by the user to control the individual influence of a measurement on the whole experiment.

The basic idea is to minimize the distances between the model function at certain time points and the corresponding measurement values. These distances are called the residuals of the data fitting problem. In the ideal case, the residuals are zero indicating a perfect fit of the measurements by the model function.

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

$$p \in \mathbb{R}^n : \quad \begin{array}{l} \min \ \sum_{i=1}^{n_d} \ w_i (h(p,t_i) - y_i)^2 \\ p_l \leq p \leq p_u \ \ , \end{array} \tag{2}$$

where p_l and p_u are suitable lower and upper bounds for the parameters to be estimated.

Example 1.1 *We want to fit some parameters $p_1, \ldots, p_4$, so that the data of Table 1 are approximated by a rational function*

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

see Lindström [24] and Deuflhard, Apostolescu [8]. All weights are set to 1 leading to the least squares data fitting problem

$$p \in \mathbb{R}^4 : \ \min \ \sum_{i=1}^{l} \left(h(p,t_i) - y_i\right)^2 \ \ .$$

When starting the code DFNLP of Schittkowski [38] from $p^0 = (0.25, 0.39, 0.415, 0.39)^T$ with a termination tolerance of 10^{-12}, we get the solution

$$p^\star = (0.1928, 0.1938, 0.1246, 0.1370)^T$$

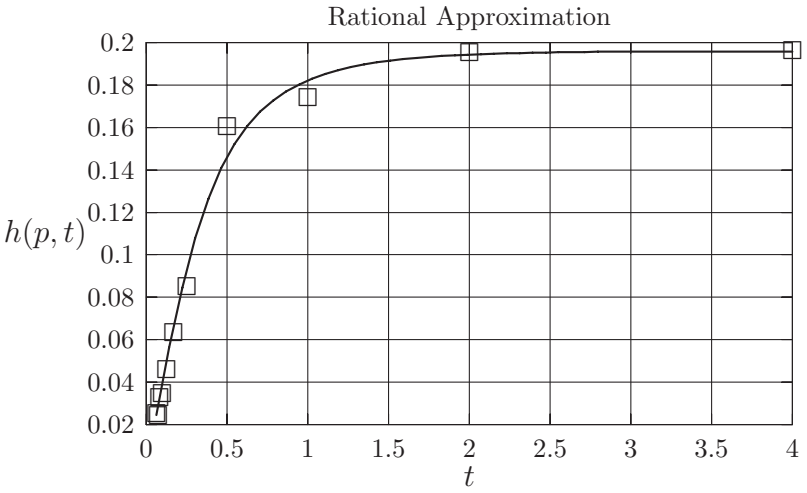

Figure 1: Function and Data Plot

after 13 iterations with a final residual norm $3.1 \cdot 10^{-4}$. *Individual residuals and relative errors are also listed in Table 1. Model function and data are plotted in Figure 1.*

Table 1: Experimental Data and Final Residuals

t_i	y_i	*error*	t_i	y_i	*error*
0.0625	0.0246	15.6 %	0.25	0.0844	10.0 %
0.0714	0.0235	2.9 %	0.5	0.16	7.0 %
0.0823	0.0323	11.9 %	1.0	0.1735	5.1 %
0.1	0.0342	3.9 %	2.0	0.1947	1.0 %
0.125	0.0456	0.2 %	4.0	0.1957	0.7 %
0.167	0.0627	0.2 %			

The report is one out of a series of Mathcad test problem collections by which numerical routines are tested and the implementation of optimization problems and dynamical systems is outlined, i.e.,

1. nonlinear programming [42],
2. ordinary differential equations [43],
3. differential algebraic equations [44],

4. partial differential equations [45],

5. partial differential algebraic equations [46].

Section 2 contains a brief outline of least squares optimality conditions which are the basis for all efficient optimization methods. The most famous one is the Gauss-Newton method which is described in Section 3 together with some of its variants. A special one based on a transformation into a general constrained optimization problem and subsequent solution by a sequential quadratic programming (SQP) method is shown in Section 4. Section 5 contains list of the Mathcad worksheet files and some further details about problem structure, practical background, and references. A detailed example is shown in Section 6 to illustrate the implementation of a least squares data fitting problem in Mathcad. An appendix contains a list of individual numerical results including final residual values, number of function calls, and number of iterations until successful termination, which have been obtained by the code DFNLP [38].

2 Optimality Conditions

The goal is to minimize the sum of squares of distances of a certain model function from experimental measurement values. However, we are not able to exploit this specific structure mathematically. Instead, we write the parameter estimation problem in the form of a least squares problem, where a sum of squared nonlinear functions is to be minimized,

$$\begin{aligned} &\min \ \tfrac{1}{2} \textstyle\sum_{i=1}^{l} \ f_i(p)^2 \\ &p \in I\!R^n \ . \end{aligned} \tag{3}$$

These problems possess a long history in mathematical programming and are extremely important in practice, particularly in nonlinear data fitting or maximum likelihood estimation. In consequence, a large number of mathematical algorithms is available for solving (3). To understand their basic features, we introduce the notation

$$F(p) = (f_1(p), \ldots, f_l(p))^T$$

for the objective function vector, and let $f(p) = \frac{1}{2} \sum_{i=1}^{l} \ f_i(p)^2$. Then

$$\nabla f(p) = \nabla F(p) F(p) \tag{4}$$

defines the Jacobian of the objective function with

$$\nabla F(p) = (\nabla f_1(p), \ldots, \nabla f_l(p)) \ .$$

If we assume now that all functions $f_1, \ldots, f_l$ are twice continuously differentiable, we get the Hessian matrix of f

$$\nabla^2 f(p) = \nabla F(p) \nabla F(p)^T + B(p) \quad , \tag{5}$$

where

$$B(p) = \sum_{i=1}^{l} f_i(p) \nabla^2 f_i(p) \quad . \tag{6}$$

Then we derive the following necessary optimality criteria.

Theorem 2.1 *Let f be twice continuously differentiable, and $p^\star$ a local solution of the least squares problem (3). Then*

a) $\nabla F(p^\star) F(p^\star) = 0$,

b) $\nabla F(p^\star) \nabla F(p^\star)^T + B(p^\star)$ *is positive semi-definite.*

For a parameter estimation problem with an ideal fit where model function values coincide with experimental data, we get $f_i(p^\star) = 0$ and condition a) trivially holds. In this case, condition b) is equivalent to the requirement that $\nabla F(p^\star)$ possesses full rank. In a very similar way, a sufficient optimality condition can be formulated.

The notation and the optimality condition is to be motivated by an example that was frequently used in the past to test unconstrained minimization algorithms and that is known under the name *banana function*, see Rosenbrock [35] or Schittkowski [37].

Example 2.1 *We want to minimize the sum of squares of two functions of the form*

$$f(p_1, p_2) = 100(p_2 - {p_1}^2)^2 + (1 - p_1)^2 = \frac{1}{2} \left(200(p_2 - {p_1}^2)^2 + 2(1 - p_1)^2 \right)$$

where

$$F(p_1, p_2) = \sqrt{2} \left(10(p_2 - {p_1}^2), 1 - p_1 \right)^T \quad ,$$

see Figure 2 for a surface plot. It is easy to see that $p^\star = (1, 1)^T$ is the unique optimal solution of the least squares problem. With the notation introduced above, the optimality condition

$$\begin{aligned} \nabla f(p_1^\star, p_2^\star) &= \nabla F(p_1^\star, p_2^\star) F(p_1^\star, p_2^\star) \\ &= 2 \begin{pmatrix} -200 p_1^\star (p_2^\star - {p_1^\star}^2) + p_1^\star - 1 \\ 100 (p_2^\star - {p_1^\star}^2) \end{pmatrix} \\ &= 0 \end{aligned}$$

is satisfied and the matrix

$$\begin{aligned}\nabla^2 f(p_1^\star, p_2^\star) &= \nabla F(p_1^\star, p_2^\star)\nabla F(p_1^\star, p_2^\star)^T + B(p_1^\star, p_2^\star) \\ &= 2\begin{pmatrix} 600p_1^{\star 2} - 200p_2^\star + 1 & -200p_1^\star \\ -200p_1^\star & 100 \end{pmatrix} \\ &= 2\begin{pmatrix} 401 & -200 \\ -200 & 100 \end{pmatrix}\end{aligned}$$

is positive definite.

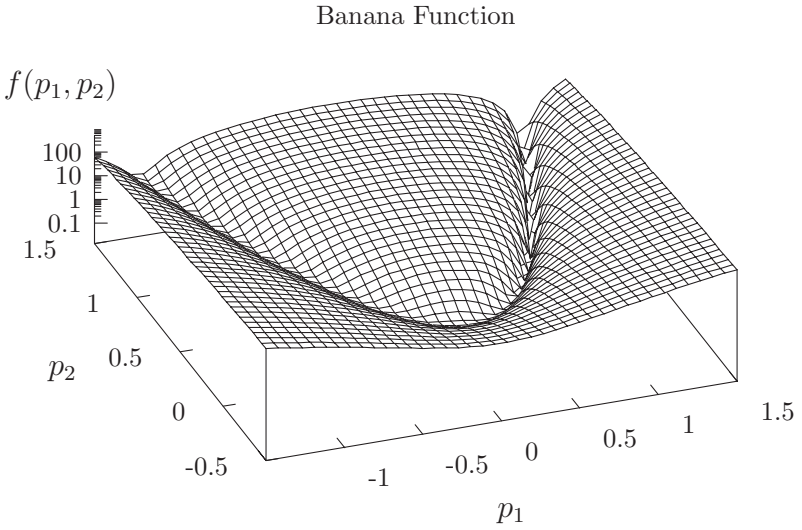

Figure 2: Surface Plot

3 Gauss-Newton and Related Methods

Proceeding from a given iterate p_k, Newton's method can be applied to (3) to get a search direction $d_k \in I\!R^n$ by solving the linear system

$$\nabla^2 f(p_k)d + \nabla f(p_k) = 0$$

or, alternatively,

$$\nabla F(p_k)\nabla F(p_k)^T d + B(p_k)d + \nabla F(p_k)F(p_k) = 0 \ . \tag{7}$$

Assume that

$$F(p^\star) = (f_1(p^\star), \ldots, f_l(p^\star))^T = 0$$

at an optimal solution $p^\star$. Then we neglect matrix $B(p_k)$ in (7), see also (6), and (7) defines the so-called normal equations of the linear least squares problem

$$\begin{aligned} &\min \ \|\nabla F(p_k)^T d + F(p_k)\| \\ &d \in I\!R^n \ . \end{aligned} \tag{8}$$

A new iterate is obtained by $p_{k+1} = p_k + \alpha_k d_k$, where d_k is a solution of (8) and where α_k denotes a suitable steplength parameter. It is obvious that a quadratic convergence rate is achieved when starting sufficiently close to an optimal solution. The above calculation of a search direction is known as the Gauss-Newton method and represents the traditional way to solve nonlinear least squares problems, see Björck [3] for more details.

In general, the Gauss-Newton method possesses the attractive feature that it converges quadratically although we do not provide any second order information. A typical theorem that is found in any textbook about numerical analysis can be formulated in the following way.

Theorem 3.1 *Assume that the unconstrained least squares problem (3) has an optimal solution $p^\star$ with $F(p^\star) = 0$, and that the Jacobian matrix of F possesses full rank and is Lipschitz continuous in a neighborhood of $p^\star$. If the starting point p_0 of the Gauss-Newton method is sufficiently close to $p^\star$, then the iterates p_k with steplength $\alpha_k = 1$ converge quadratically to $p^\star$, i.e., there is a positive constant γ with*

$$\|p_{k+1} - p^\star\| \leq \ \gamma \, \|p_k - p^\star\|^2$$

for all k.

Lipschitz continuity of the Jacobian matrix of F is a bit stronger than usual continuity and is defined by

$$\|\nabla F(p) - \nabla F(q)\| \leq L \, \|p - q\|$$

for all p and q in a neighborhood of $p^\star$, where L is a suitable constant.

Example 3.1 *The test example is the banana function of Rosenbrock that is also considered in the previous section, where we minimize*

$$f(p_1, p_2) = 100(p_2 - {p_1}^2)^2 + (1 - p_1)^2 \ .$$

To perform a Gauss-Newton step, we have to solve the following system of linear equations, the so-called normal equations, where p_1 and p_2 denote the actual iterate,

$$\begin{pmatrix} 400{p_1}^2 + 1 & -200p_1 \\ -200p_1 & 100 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} + \begin{pmatrix} -200p_1(p_2 - {p_1}^2) + p_1 - 1 \\ 100(p_2 - {p_1}^2) \end{pmatrix} = 0 \ .$$

Now assume that $p = (0,0)^T$ *is the initial iterate of the Gauss-Newton algorithm. Then* $d = (1,0)^T$ *is solution of the above system, i.e., of*

$$\begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = 0 \quad .$$

The next iterate is

$$\overline{p} = p + d = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad .$$

We have to solve again the normal equations

$$\begin{pmatrix} 401 & -200p_1 \\ -200 & 100 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} + \begin{pmatrix} 200 \\ -100 \end{pmatrix} = 0$$

from which we get $\overline{d} = (0,1)^T$ *and the second iterate*

$$p^\star = \overline{p} + \overline{d} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad .$$

We get exactly the optimal solution in only two steps, an accidental situation.

However, the assumptions of the convergence theorem of Gauss-Newton methods are very strong and cannot be satisfied in real situations. We have to expect difficulties in case of non-zero residuals, rank-deficient Jacobian matrices, non-continuous derivatives, and starting points far away from a solution.

Further difficulties arise when trying to solve large residual problems, where $F(p^\star)^T F(p^\star)$ is not sufficiently small, for example relative to $\|\nabla F(p^\star)\|$. Numerous proposals have been made in the past to deal with this situation, and it is outside the scope of this chapter to give a review of all possible attempts developed in the last 30 years. Only a few remarks are presented to illustrate basic features of the main approaches, for further reviews see Gill, Murray and Wright [12], Ramsin and Wedin [31], or Dennis [7].

A very popular method is known under the name Levenberg-Marquardt algorithm, see Levenberg [22] and Marquardt [25]. The key idea is to replace the Hessian in (7) by a multiple of the identity matrix, say $\lambda_k I$, with a suitable positive factor λ_k. We get a uniquely solvable system of linear equations of the form

$$\nabla F(p_k) \nabla F(p_k)^T d + \lambda_k d + \nabla F(p_k) F(p_k) = 0 \quad .$$

For the choice of λ_k and the relationship to so-called *trust region methods*, see Moré [26].

A more sophisticated idea is to replace $B(p_k)$ in (7) by a quasi-Newton-matrix B_k, see Dennis [6]. But some additional safeguards are necessary to deal with indefinite matrices

$\nabla F(p_k)\nabla F(p_k)^T + B_k$ in order to get a descent direction. A modified algorithm is proposed by Gill and Murray [11], where B_k is either a second-order approximation of $B(p_k)$, or a quasi-Newton matrix. In this case, a diagonal matrix is added, $\nabla F(p_k)\nabla F(p_k)^T + B_k$, to obtain a positive definite matrix. Lindström [23] proposes a combination of a Gauss-Newton and a Newton method by using a certain subspace minimization technique.

If, however, the residuals are too large, there is no possibility to exploit the special structure and a general unconstrained minimization algorithm, for example a quasi-Newton method, can be applied as well.

4 Least Squares Minimization by SQP Methods

Many efficient special purpose computer programs are available to solve unconstrained nonlinear least squares problems. On the other hand, there exists a very simple approach to combine the valuable properties of Gauss-Newton methods with that of SQP algorithms in a straightforward way with almost no additional efforts.

We proceed from an unconstrained least squares problem in the form

$$\begin{array}{l} \min \frac{1}{2} \sum_{i=1}^{l} \; f_i(p)^2 \\ p \in I\!R^n \quad , \end{array} \tag{9}$$

see also (3). Since most nonlinear least squares problems are ill-conditioned, it is not recommended to solve (9) directly by a general nonlinear programming method. But we will see in this section that a simple transformation of the original problem and its subsequent solution by a SQP method retains typical features of a special purpose code and prevents the need to take care of negative eigenvalues of an approximated Hessian matrix as in the case of alternative approaches. The corresponding computer program can be implemented in a few lines provided that a SQP algorithm is available.

The transformation, also described in Schittkowski [38], consists of introducing l additional variables $z = (z_1, \ldots, z_l)^T$ and l additional equality constraints of the form

$$f_i(p) - z_i = 0 \; , \quad i = 1, \ldots, l \quad . \tag{10}$$

Then the equivalent transformed problem is

$$(p, z) \in I\!R^{n+l} : \begin{array}{l} \min \; \frac{1}{2} z^T z \\ F(p) - z = 0 \quad , \end{array} \tag{11}$$

$F(p) = (f_1(p), \ldots, f_l(p))^T$. We consider now (11) as a general nonlinear programming problem of the form

$$\bar{p} \in I\!R^{\bar{n}} : \begin{array}{l} \min \bar{f}(\bar{p}) \\ \bar{g}(\bar{p}) = 0 \end{array} \tag{12}$$

with $\bar{n} = n + l$, $\bar{p} = (p, z)$, $\bar{f}(p, z) = \frac{1}{2} z^T z$, $\bar{g}(p, z) = F(p) - z$, and apply a sequential quadratic programming (SQP) method as for example described in Spellucci [48]. Sequential quadratic programming methods are the standard general purpose algorithms for solving smooth nonlinear optimization problems and are widely used for solving practical optimization problems. The key idea is to approximate also second order information to get a fast final convergence speed. Thus, we define a quadratic approximation of the Lagrangian function and an approximation of its Hessian matrix by a quasi-Newton matrix $\bar{B}_k$. The quadratic programming subproblem is

$$\bar{d} \in I\!R^{\bar{n}} : \begin{array}{l} \min \frac{1}{2} \bar{d}^T \bar{B}_k \bar{d} + \nabla \bar{f}(\bar{p}_k)^T \bar{d} \\ \nabla \bar{g}(\bar{p}_k)^T \bar{d} + \bar{g}(\bar{p}_k) = 0 \ , \end{array} \tag{13}$$

where a bar is used to avoid confusion with the notation of the previous section. In (13), $\bar{p}_k = (p_k, z_k)$ is a given iterate and

$$\bar{B}_k = \left(\begin{array}{ccc} B_k & : & C_k \\ C_k^T & : & D_k \end{array} \right) \tag{14}$$

with $B_k \in I\!R^{n \times n}$, $C_k \in I\!R^{n \times l}$, and $D_k \in I\!R^{l \times l}$, a given approximation of the Hessian of the Lagrangian function $L(\bar{p}, u)$ defined by

$$\begin{aligned} L(\bar{p}, u) &= \bar{f}(\bar{p}) - u^T \bar{g}(\bar{p}) \\ &= \tfrac{1}{2} z^T z - u^T (F(p) - z) \ . \end{aligned}$$

Since

$$\nabla_{\bar{p}} L(\bar{p}, u) = \left(\begin{array}{c} -\nabla F(p) u \\ z + u \end{array} \right)$$

and

$$\nabla^2_{\bar{p}} L(\bar{p}, u) = \left(\begin{array}{ccc} B(p, u) & : & 0 \\ 0 & : & I \end{array} \right)$$

with

$$B(p, u) = -\sum_{i=1}^{l} u_i \nabla^2 f_i(p) \ , \tag{15}$$

it seems to be reasonable to proceed now from a quasi-Newton matrix given by

$$\bar{B}_k = \left(\begin{array}{ccc} B_k & : & 0 \\ 0 & : & I \end{array} \right) \ , \tag{16}$$

where $B_k \in I\!R^{n \times n}$ is a suitable positive definite approximation of $B(p_k, u_k)$. Insertion of this $\bar{B}_k$ into (13) leads to the equivalent quadratic programming subproblem

$$(d, e) \in I\!R^{n+l} : \begin{array}{l} \min \ \frac{1}{2} d^T B_k d + \frac{1}{2} e^T e + z_k^T e \\ \nabla F(p_k)^T d - e + F(p_k) - z_k = 0 \ , \end{array} \tag{17}$$

where we replace $\bar{d}$ by (d, e). Some simple calculations show that the solution of the above quadratic programming problem is identified by the linear system

$$\nabla F(p_k)\nabla F(p_k)^T d + B_k d + \nabla F(p_k)F(p_k) = 0 \quad . \tag{18}$$

This equation is identical to (7), if $B_k = B(p_k)$, and we obtain the following theorem.

Theorem 4.1 *Assume that for a given iterate $p_k \in I\!R^n$, an SQP step is performed with $B_k = B(p_k)$, $B(p)$ defined by (6) and $\bar{B}_k$ decomposed in the form (16). Then we obtain a Newton direction for solving the unconstrained least squares problem (9).*

Note that $B(p)$ defined by (6) and $B(p)$ defined by (15) coincide at an optimal solution of the least squares problem, since $F(p_k) + z_k = -u_k$. Based on the above considerations, an SQP method can be applied to solve (11) directly. The quasi-Newton-matrices $\bar{B}_k$ are always positive definite, and consequently also the matrix B_k defined by (14). Therefore, we omit numerical difficulties imposed by negative eigenvalues as found in the usual approaches for solving least squares problems.

When starting the SQP method, one could proceed from a user-provided initial guess p_0 for the variables and define

$$\begin{aligned} z_0 &= F(p_0) \ , \\ B_0 &= \begin{pmatrix} \mu I & : & 0 \\ 0 & : & I \end{pmatrix} \ , \end{aligned} \tag{19}$$

guaranteeing a feasible starting point $\bar{p}_0$. The choice of B_0 is of the form (16) and allows a user to provide some information on the estimated size of the residuals, if available. If it is known that the final norm $F(p^\star)^T F(p^\star)$ is close to zero at the optimal solution $p^\star$, the user could choose a small μ in (19). At least in the first iterates, the search directions are similar to a Gauss-Newton direction. Otherwise, a user could define $\mu = 1$, if a large residual is expected.

Example 4.1 *We consider again the banana function*

$$f(p_1, p_2) = 100(p_2 - {p_1}^2)^2 + (1 - p_1)^2 \quad .$$

When applying the nonlinear programming code NLPQL of Schittkowski [36], an implementation of a general purpose SQP method, we get the iterates of Table 2 starting at $p_0 = (-1.2, 1.0)^T$. The last column contains an internal stopping condition based on the optimality criterion, in our unconstrained case equal to

$$|\nabla f(p_k) B_k^{-1} \nabla f(p_k)|$$

with a quasi-Newton matrix B_k. We observe a very fast final convergence speed, but a relatively large number of iterations. If we omit now the factor $\frac{1}{2}$ for simplicity, the equivalent constrained nonlinear programming problem is

$$\begin{aligned} & \min \ {z_1}^2 + {z_2}^2 \\ p_1, p_2, z_1, z_2 : \ & 10(p_2 - {p_1}^2) - z_1 = 0 \ , \\ & 1 - p_1 - z_2 = 0 \ . \end{aligned}$$

When using now the same algorithm NLPQL, we get the results of Table 3. In this case, the last column contains the stopping condition based on Theorem 2.1. Obviously, the convergence speed is much faster.

Table 2: NLP Formulation of Banana Function

k	$f(p_k)$	$s(p_k)$	k	$f(p_k)$	$s(p_k)$
0	24.20	$0.54 \cdot 10^{5}$	29	$0.15 \cdot 10^{-2}$	$0.18 \cdot 10^{-2}$
1	12.21	$0.12 \cdot 10^{3}$	30	$0.39 \cdot 10^{-3}$	$0.57 \cdot 10^{-3}$
2	2.547	$0.19 \cdot 10^{1}$	31	$0.36 \cdot 10^{-4}$	$0.62 \cdot 10^{-4}$
3	2.391	$0.57 \cdot 10^{-1}$	32	$0.13 \cdot 10^{-5}$	$0.26 \cdot 10^{-5}$
4	2.346	$0.86 \cdot 10^{-1}$	33	$0.11 \cdot 10^{-7}$	$0.24 \cdot 10^{-7}$
5	1.942	$0.23 \cdot 10^{-1}$	34	$0.75 \cdot 10^{-10}$	$0.15 \cdot 10^{-9}$
...	...	...	35	$0.15 \cdot 10^{-15}$	$0.31 \cdot 10^{-15}$

Table 3: Least Squares Formulation of Banana Function

k	$f(p_k)$	$s(p_k)$	k	$f(p_k)$	$s(p_k)$
0	24.20	$0.82 \cdot 10^{2}$	6	$0.32 \cdot 10^{-4}$	$0.61 \cdot 10^{-4}$
1	22.21	$0.41 \cdot 10^{2}$	7	$0.29 \cdot 10^{-6}$	$0.57 \cdot 10^{-6}$
2	5.120	$0.11 \cdot 10^{1}$	8	$0.37 \cdot 10^{-9}$	$0.73 \cdot 10^{-9}$
3	$0.42 \cdot 10^{-1}$	$0.74 \cdot 10^{-1}$	9	$0.42 \cdot 10^{-13}$	$0.84 \cdot 10^{-13}$
4	$0.12 \cdot 10^{-1}$	$0.17 \cdot 10^{-1}$	10	$0.28 \cdot 10^{-17}$	$0.56 \cdot 10^{-17}$
5	$0.17 \cdot 10^{-2}$	$0.31 \cdot 10^{-2}$			

5 The Test Problems

The subsequent table contains a list of all test problems together with the number of parameters to be estimated n_p, the number of experimental data n_d, a brief description of the practical or mathematical background, and some references. The model equations have first been implemented in the modelling language PCOMP, see Dobmann et al. [9] or Schittkowski [39, 40, 41]. The transformation into Mathcad worksheets follows a unified format based on the PCOMP equations. Thus, the implementations do not exploit all possible features of Mathcad to get the most elegant and compact description. All mcd-files can be downloaded from the home page of the author[4].

Data Fitting Problems

name	n_p	n_d	*background*	*ref*
2VALLEYS	2	4	Academic test problem with two local minima	[47]
APPRX3	6	10	Rational approximation of data	[2]
ATROP_EX	4	24	Atropin-chase binding, linear model	
BENNETT5	3	154	Superconductivity magnetization modeling (NIST study)	
BOXBOD	2	6	Biochemical oxygen demand (NIST study)	[4]
CAT_SEP	5	5	Catalysator separation problem	
CHWIRUT1	3	215	Ultrasonic reference block (NIST study)	
CHWIRUT2	3	54	Ultrasonic reference block (NIST study)	
DANWOOD	2	6	Energy radiated from a carbon filament lamp (NIST study)	[5]
DFE1	8	9	Explicit test function with local solutions, cycling model function etc.	
DOAS	21	100	Differential optical spectral absorption	
E_FIT	3	12	Rational-exponential data fitting	
ECKERLE4	3	35	Circular interference transmittance (NIST study)	
ELA_TUBX	3	40	Waves propagating in a liquid-filled elastic tube (KDVB equation)	[17], [50]
ENZREAC	4	13	Enzyme reaction, rational approximation	
EW_WAVEX	2	48	Wave propagation in media with nonlinear steepening and dispersion	[13]

(continued)

[4] http://www.klaus-schittkowski.de

name	n_p	n_d	*background*	*ref*
EXP_FIT1	2	28	Exponential data fitting	
EXP_FIT2	7	33	Exponential data fitting	
EXP_FIT3	2	27	Exponential data fitting	
EXP_FIT4	5	19	Exponential data fitting	
EXP_FIT6	2	4	Exponential data fitting	
EXP_SMPL	2	81	Single term exponential model, large errors in data	
EXP2TERM	5	20	Two-exponential model	
GAMMAS	7	27	Analysis of a gamma spectrum	
GAUSS1	8	250	Two well-separated Gaussians (NIST study)	
GAUSS2	8	250	Two slightly-blended Gaussians (NIST study)	
GAUSS3	8	250	Two strongly-blended Gaussians (NIST study)	
GLU_RATE	4	13	In-vivo glucose turnover rate	
HAHN1	7	236	Thermal expansion of copper (NIST study)	
HEAT_XX	2	99	Linear diffusion with constant parameters, exact solution	
ILL_COND	100	100	Ill-conditioned test function, many parameters	
INFINITE	3	21	Infinitely many solutions	
INTEG_X	3	25	Population dynamics	[30]
KIRBY2	5	151	Scanning electron microscope (NIST study)	
LANCZOS1	6	24	Exponential nonlinear regression (NIST study)	[21]
LANCZOS2	6	24	Exponential nonlinear regression (NIST study)	[21]
LANCZOS3	6	24	Exponential nonlinear regression (NIST study)	[21]
LIN_HC_X	3	165	Linear heat conduction	[1]
MAC_ECO	6	186	Macroeconomic time series of currency notes in circulation	[49]
MARKET	7	100	Dynamic economic market	
MGH09	4	11	Rational nonlinear regression (NIST study)	[27]
MGH10	3	16	Exponential nonlinear regression (NIST study)	[27]
MGH17	5	33	Exponential nonlinear regression (NIST study)	[27], [29]
MICHMENT	2	12	Michaelis-Menten kinetics	[47], [52]
MISRA1A	2	14	Monomolecular adsorption (NIST study)	
MISRA1B	2	14	Monomolecular adsorption (NIST study)	
MISRA1C	2	14	Monomolecular adsorption (NIST study)	
MISRA1D	2	14	Monomolecular adsorption (NIST study)	
MIX_PAT2	3	33	Mixing pattern inside a polymerization reactor	
MIX_PAT3	1	27	Mixing pattern inside a polymerization reactor	
MIX_PAT4	3	28	Mixing pattern inside a polymerization reactor	

(continued)

name	n_p	n_d	*background*	*ref*
MONOD	4	10	Monod-Wymnan-Changeux kinetic equation	[47], [33]
MORTALTY	2	9	Mortality rate by Gompertz function	[15]
NELSON	3	128	Analysis of performance degradation data (NIST study)	[28]
OSCILL_S	16	50	Oscillating system with exact known solution	[53]
OSCILL_X	16	50	Oscillating system	[53]
PARID15	3	16	Parameter identification model, 15 normally distributed experimental values	
PARID30	3	31	Parameter identification model, 30 normally distributed experimental values	
PARID60	3	61	Parameter identification model, 60 normally distributed experimental values	
PARID120	3	121	Parameter identification model, 120 normally distributed experimental values	
RAD_TRAC	3	8	Radioactive tracer in two human body compartments	
RAMAN	2	101	Raman intensity of anisotrope probes	[20]
RAT42	3	9	Pasture yield with sigmoidal growth curve (NIST study)	[32]
RAT43	4	15	Dry weight of onion bulbs and tops (NIST study)	[32]
RAT_APP	4	11	Rational approximation with constraints	[24]
RAT_FIT	4	11	Fitting a rational function	[18]
RICH_GR	3	9	Richards growth model	[34]
ROSZMAN1	4	25	Quantum defects in iodine atoms (NIST study)	
RTD	2	36	Residence time distribution	
SMOOTHNG	3	500	Data smoothing	
STEP_RES	3	22	Second-order equation with dead time and step response data	[51]
SULFATE	4	9	Compartmental analysis in humans with radioactive sulfate as tracer	[47]
THERMRES	3	10	Thermistor resistance, exponential data fitting	
THURBER	7	37	Semiconductor electron mobility (NIST study)	
TIME_ACT	2	9	Time activities	
TP25	3	99	Academic test problem, highly unstable	[14]
TP203	2	3	Simple data fitting problem	[37]
TP205	2	3	Least squares problem with three terms	[37]
TP242	3	10	Exponential test function	[37]
TP244	3	10	Exponential test function	[37]
TP267	5	11	Exponential test function	[37]

(continued)

name	n_p	n_d	*background*	*ref*
TP272	6	13	Exponential test function	[37]
TP307	2	10	Exponential data fitting	[37]
TP327	2	45	Constrained exponential data fitting	[37]
TP333	3	8	Exponential data fitting	[37]
TP334	3	15	Exponential data fitting	[37]
TP350	4	6	Rational approximation	[37]
TP358	5	20	Exponential data fitting test function	[37]
TP379	11	65	Test problem of Osborne, four exponential terms	[37], [29]
TREND	6	501	Trend curve	
TRIG_APP	2	19	Trigonometric approximation for computing axial forces	
TUBTANK	1	19	Comparison of tank and tubular reactors steady state	[16]
VAPOR	2	11	Vapor-liquid equilibrium	[10]
VISC_ELA	10	24	Memory function of visco-elastic substances	
WEIBULL	2	14	Weibull distribution	

6 A Mathcad Worksheet Example

Mathcad[5] is an interactive GUI with a large number of built-in mathematical functions. Special commands allow to solve also constrained nonlinear programming problems. The subsequent lines describe the usage of these functions and are taken from the Mathcad documentation:

Least Squares Data Fitting

```
LeastSquaresFit(vx, vy, F, guess, conf, [Stdy], [LBUB], [Acc])
```

Takes real vectors `vx` and `vy` of identical length, a fitting function `F(x,b)` of one variable with an arbitrary number of parameters, `b`, a guess vector with one guess value for each parameter, the desired confidence limit `conf`, a percentage between 0 and 1, inclusive. An additional optional vector of standard deviations, `Stdy`, the same length as `vy`, an optional two-column matrix of lower and upper bounds on the parameters, `LBUB`, with the same number of rows as guess; and an optional convergence accuracy `Acc`, may be specified. Returns a three column matrix, where the first column contains the values for the fitted parameters, and the second and third columns contain the left and right boundaries, respectively, for the confidence interval.

Notes:

- Number of parameters cannot exceed length of `vx`.

[5] http://www.mathcad.com

- Define the function `F` before using it in `LeastSquaresFit`. It is possible to specify the function `F(x,b)` with `b` as a vector, or with a series of scalar variables in the function's argument list.
- When specifying the function in `LeastSquaresFit`, use only the function name, not its arguments.
- If you are using more than one of the optional arguments, they must be specified in their relative order shown above.
- The default value for `Acc` is 10^{-7}.

Example 6.1 *To give an impression how a test problem is implemented, we consider the rational data fitting example 1.1, where we want to fit the parameters $b_1, \ldots, b_4$, so that the data of Table 1 are approximated by the rational function*

$$F(x,b) = b_1 \frac{x^2 + b_2 x}{x^2 + b_3 x + b_4} \quad ,$$

see Lindström [24] and Deuflhard, Apostolescu [8]. Subsequently, the Mathcad implementation is listed, see Figure 3. The mcd-files contain not only data and functions which describe the optimization problem, but also the computed solution together with confidence intervals and the final residual, see Figure 4.

RAT_APP

Data Fitting

Description: Rational approximation

Reference: Lindstrom P. (1983): A general purpose algorithm for nonlinear least squares problems with nonlinear constraints, Report UMINF-103.83, Institute of Information Processing, University of Umea, Umea, Sweden

Model Function:

$$F(x,b) := \frac{\left(x^2 + b_2 \cdot x\right) \cdot b_1}{x^2 + b_3 \cdot x + b_4}$$

Experimental Data:

$$vx := \begin{pmatrix} 0.0625 \\ 0.0714 \\ 0.0823 \\ 0.1 \\ 0.125 \\ 0.167 \\ 0.25 \\ 0.5 \\ 1 \\ 2 \\ 4 \end{pmatrix} \qquad vy := \begin{pmatrix} 0.0246 \\ 0.0235 \\ 0.0323 \\ 0.0342 \\ 0.0456 \\ 0.0627 \\ 0.0844 \\ 0.16 \\ 0.1735 \\ 0.1947 \\ 0.1957 \end{pmatrix}$$

Starting Values:

$$guess := \begin{pmatrix} 0.25 \\ 0.39 \\ 0.415 \\ 0.39 \end{pmatrix}$$

Figure 3: Mathcad Declarations

Minimize:

$$\text{Res} := \text{LeastSquaresFit}(vx, vy, F, \text{guess}, 0.99)$$

Computed Solution:

$$\text{Res} = \begin{pmatrix} 0.192769 & 0.192619 & 0.192918 \\ 0.19399 & 0.191404 & 0.196576 \\ 0.124739 & 0.123676 & 0.125803 \\ 0.137002 & 0.13582 & 0.138185 \end{pmatrix}$$

$$b := \text{Res}^{\langle 1 \rangle}$$

$$\sum_{i=1}^{11} \left(F(vx_i, b) - vy_i\right)^2 = 3.104228 \times 10^{-4}$$

Plot:

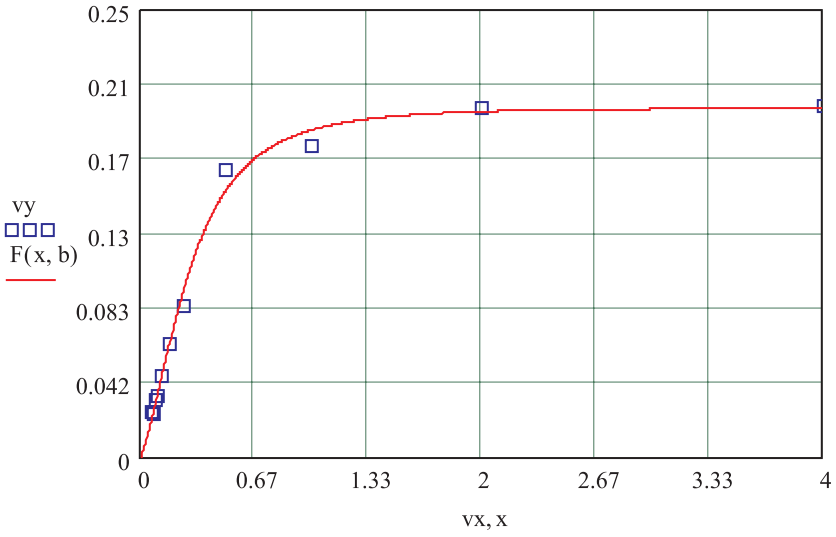

Figure 4: Mathcad Results

Acknowledgement: The author would like to thank Tobias Kreisel, Sebastian Götschel, Michael Manger, Thomas Feulner, and Nikolas Tautenhahn for preparing the Mathcad worksheets.

References

[1] Adjerid S., Flaherty J.E. (1986): *A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations*, SIAM Journal on Numerical Analysis, Vol. 23, 778-796

[2] Bartholomew-Biggs M.C. (1995): *Implementing and using a FORTRAN 90 version of a subroutine for non-linear least squares calculations*, Report, NOC, Hatfield

[3] Björck A. (1990): *Least Squares Methods,* Elsevier, Amsterdam

[4] Box G.P., Hunter W.G., Hunter J.S. (1978): *Statistics for Experimenters*, John Wiley, New York

[5] Daniel C., Wood F.S. (1980): *Fitting Equations to Data*, John Wiley, New York

[6] Dennis J.E.jr. (1973): *Some computational technique for the nonlinear least squares problem,* in: Numerical Solution of Systems of Nonlinear Algebraic Equations, G.D. Byrne, C.A. Hall eds., Academic Press, New York, London

[7] Dennis J.E.jr. (1977): *Nonlinear least squares,* in: The State of the Art in Numerical Analysis, D. Jacobs ed., Academic Press, New York, London

[8] Deuflhard P., Apostolescu V. (1977): *An underrelaxed Gauß-Newton method for equality constrained nonlinear least squares,* Proceedings of the IFIP Conference on Optimization Techniques, Part 2, A.V. Balakrishnan, Thoma M. eds., Lecture Notes in Control and Information Sciences, Vol. 7, 22-32, Springer, Berlin

[9] Dobmann M., Liepelt M., Schittkowski K. (1995): *Algorithm 746: PCOMP: A Fortran code for automatic differentiation,* ACM Transactions on Mathematical Software, Vol. 21, No. 3, 233-266

[10] Edgar T.F., Himmelblau D.M. (1988): *Optimization of Chemical Processes,* McGraw Hill, New York

[11] Gill P.E., Murray W. (1978): *Algorithms for the solution of the non-linear least-squares problem,* SIAM Journal on Numerical Analysis, Vol. 15, 977-992

[12] Gill P.E., Murray W., Wright M.H. (1981): *Practical Optimization,* Academic Press, New York, London

[13] Hamdi S., Gottlieb J.J., Hanson J.S. (2001): *Numerical solutions of the equal width wave equation using an adaptive method of lines*, in: Adaptive Methods of Lines, A. Vande Wouwer, Ph. Saucec Ph., W. Schiesser eds., Chapman and Hall/CRC, Boca Raton

[14] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer-Verlag

[15] Hooker P.F. (1965): *Benjamin Gompertz*, Journal of the Institute of Actuaries, Vol. 91, 203-212

[16] Ingham J., Dunn I.J., Heinzle E., Prenosil J.E. (1994): *Chemical Engineering Dynamics*, VCH, Weinheim

[17] Johnson R.S. (1970): *A nonlinear equation incorporating damping and dispersion*, Journal of Fluid Dynamics, Vol. 42, 49-60

[18] Kowalik J. (1967): *A note on nonlinear regression analysis*, Australian Computational Journal, Vol. 1, 51-53

[19] Kuhn H.W., Tucker A.W. (1951): *Nonlinear programming*, in: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Programming, University of California Press, Berkeley

[20] Lagugne-Labarthet F., Bruneel J.L., Sourisseau C., Huber M.R., Börger V., Menzel H. (2001): *A microspectrometric study of the azobenzene chromophore orientation in a holographic diffraction grating inscribed on a p(HEMA-co-MMA) functionalized copolymer film*, Journal of Raman Spectroscopy, Vol. 32, 665-675

[21] Lanczos C. (1956): *Applied Analysis*, Prentice Hall, Englewood Cliffs

[22] Levenberg K. (1944): *A method for the solution of certain problems in least squares,* Quarterly Applied Mathematics, Vol. 2, 164-168

[23] Lindström P. (1982): *A stabilized Gauß-Newton algorithm for unconstrained least squares problems,* Report UMINF-102.82, Institute of Information Processing, University of Umea, Umea, Sweden

[24] Lindström P. (1983): *A general purpose algorithm for nonlinear least squares problems with nonlinear constraints,* Report UMINF-103.83, Institute of Information Processing, University of Umea, Umea, Sweden

[25] Marquardt D. (1963): *An algorithm for least-squares estimation of nonlinear parameters,* SIAM Journal of Applied Mathematics, Vol. 11, 431-441

[26] Moré J.J. (1977): *The Levenberg-Marquardt algorithm: implementation and theory,* in: Numerical Analysis, G. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer, Berlin

[27] Moré J.J., Garbow B.S., Hillstrom K.E. (1981): *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, Vol. 7, No. 1, 17-41

[28] Nelson W. (1981): *Analysis of performance-degradation data*, IEEE Transactions on Reliability, Vol. 2, No. 2, 149-155

[29] Osborne M.R. (1972): *Some aspects of nonlinear least squares calculations*, in: Numerical Methods for Nonlinear Optimization, F. Lootsma ed., Academic Press, New York

[30] Pennington S.V., Berzins M. (1994): *New NAG Library software for first-order partial differential equations*, ACM Transactions on Mathematical Software, Vol. 20, No. 1, 63-99

[31] Ramsin H., Wedin P.A. (1977): *A comparison of some algorithms for the nonlinear least squares problem,* Nordisk Tidstr. Informationsbehandlung (BIT), Vol. 17, 72-90

[32] Ratkowsky D.A. (1988): *Nonlinear Regression Modeling*, Marcel Dekker, New York

[33] Reich J.G.,. Zinke I. (1974): *Analysis of kinetic and binding measurements, IV Redundancy of model parameters*, Studia Biophysics, Vol. 43, 91-107

[34] Richter O., Söndgerath D. (1990): *Parameter Estimation in Ecology*, VCH, Weinheim

[35] Rosenbrock H.H. (1969): *An automatic method for finding the greatest and least value of a function,* Computer Journal, Vol. 3, 175-183

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

[37] Schittkowski K. (1987): *More Test Examples for Nonlinear Programming*, Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer-Verlag

[38] Schittkowski K. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method,* in: Trends in Mathematical Optimization, K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, Boston, Basel, Berlin, 295-309

[39] Schittkowski K. (2002): *EASY-FIT: A software system for data fitting in dynamic systems*, Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169

[40] Schittkowski K. (2002): *Test problems for nonlinear programming - user's guide*, Report, Department of Mathematics, University of Bayreuth

[41] Schittkowski K. (2004): *PCOMP: A modeling language for nonlinear programs with automatic differentiation*, in: *Modeling Languages in Mathematical Optimization*, J. Kallrath ed., Kluwer, Norwell, MA, 349-367

[42] Schittkowski K. (2004): *110 Mathcad worksheets for nonlinear programming*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[43] Schittkowski K. (2004): *295 Mathcad worksheets for differential equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[44] Schittkowski K. (2004): *28 Mathcad worksheets for differential algebraic equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[45] Schittkowski K. (2004): *131 Mathcad worksheets for partial differential equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[46] Schittkowski K. (2004): *17 Mathcad worksheets for partial differential algebraic equations*, Report, Dept. of Computer Science, University of Bayreuth, Germany

[47] Seber G.A.F., Wild C.J. (1989): *Nonlinear Regression*, John Wiley, New York

[48] Spellucci P. (1993): *Numerische Verfahren der nichtlinearen Optimierung*, Birkhäuser

[49] Stortelder W.J.H. (1998): *Parameter estimation in nonlinear dynamical systems*, Dissertation, National Research Institute for Mathematics and Computer Science, University of Amsterdam

[50] Vande Wouwer A., Saucec Ph., Schiesser W.E. (2001): *Adaptive Methods of Lines*, Chapman and Hall/CRC, Boca Raton

[51] Walas S.M. (1991): *Modeling with Differential Equations in Chemical Engineering*, Butterworth-Heinemann, Boston

[52] Watts D.G. (1981): *An introduction to nonlinear least squares*, in: Kinetic Data Analysis: Design and Analysis of Enzyme and Pharmacokinetic Experiments, L. Endrenyi ed., Plenum Press, New York, 1-24

[53] Zschieschang, T. Dresig, H. (1998): *Zur Zeit-Frequenz-Analyse von Schwingungen in Antrieben von Verarbeitungsmaschinen*, Fortschrittsberichte VDI, Nr. 1416, VDI, Düsseldorf, 489-506

APPENDIX: Individual Results

To show how efficiently these problems can be solved, we present some numerical performance data, i.e., number of function calls, number of iterations, and final residuals. With the default tolerances given, all problems can be solved successfully by the code DFNLP [36]. Derivatives are computed by a five-point-difference formula and termination tolerance is set to 10^{-7}. More details about the test environment and evaluation of successful returns are found in [40]. The subsequent table contains a list of all test problems with the data

name	test problem number,
n_f	number of objective function evaluations,
n_g	number of gradient evaluations of objective function,
residual	final residual.

name	n_f	n_g	*residual*
2VALLEYS	10	9	.934
APPRX3	21	21	.00538
ATROP_EX	29	26	.00405
BENNETT5	17	13	.000524
BOXBOD	10	10	1170
CAT_SEP	88	52	.00204
CHWIRUT1	11	9	2380
CHWIRUT2	16	10	513

(continued)

name	n_f	n_g	*residual*
DANWOOD	6	6	.00432
DFE1	12	12	.00853
DOAS	27	24	.000356
E_FIT	64	39	2.59
ECKERLE4	68	46	.00146
ELA_TUBX	44	27	.000853
ENZREAC	40	32	.0000192
EW_WAVEX	23	20	.000318
EXP_FIT1	21	16	.00182
EXP_FIT2	36	19	.0000105
EXP_FIT3	74	47	.0253
EXP_FIT4	18	18	.000243
EXP_FIT6	29	19	6.49E-06
EXP_SMPL	0	0	2
EXP2TERM	46	30	2.01
GAMMAS	36	31	.000794
GAUSS1	7	7	1320
GAUSS2	9	9	1250
GAUSS3	9	9	1240
GLU_RATE	18	18	6.22E-07
HAHN1	45	29	.0000276
HEAT_XX	92	50	4.24E-19
ILL_COND	2	2	1.17E-22
INFINITE	18	18	4.29E-16
INTEG_X	10	10	.000424
KIRBY2	8	8	7.97E-06
LANCZOS1	143	78	6.17E-23
LANCZOS2	120	73	1.23E-12
LANCZOS3	125	67	8.36E-10
LIN_HC_X	34	29	.000945
MAC_ECO	123	70	.000102
MARKET	10	10	2.39E-06
MGH09	410	175	.00207
MGH10	83	47	2.26E-08

(continued)

name	n_f	n_g	*residual*
MGH17	28	22	3.83E-06
MICHMENT	8	8	.0195
MISRA1A	26	18	3.77E-06
MISRA1B	45	27	2.28E-06
MISRA1C	10	9	1.24E-06
MISRA1D	7	7	1.71E-06
MIX_PAT2	31	30	.012
MIX_PAT3	13	13	.0000412
MIX_PAT4	16	16	.00693
MONOD	23	16	.808
MORTALTY	12	12	.00617
NELSON	78	51	.0154
OSCILL_S	113	88	2.35E-21
OSCILL_X	47	45	.000624
PARID15	19	16	.00286
PARID30	0	1	.00511
PARID60	26	17	.0471
PARID120	22	15	.073
RAD_TRAC	60	39	.0125
RAMAN	10	9	.00235
RAT_APP	10	9	.000413
RAT_FIT	43	37	.000307
RAT42	12	11	.000442
RAT43	72	57	.00233
RICH_GR	53	37	.00132
ROSZMAN1	6	6	.000104
RTD	19	19	.0000151
SMOOTHNG	32	32	.000167
STEP_RES	24	17	.919
SULFATE	14	14	.00594
THERMRES	24	24	.0035
THURBER	41	38	.000413
TIME_ACT	69	43	.00107
TP25	78	48	6.22E-21

(continued)

name	n_f	n_g	*residual*
TP203	7	7	1.12E-16
TP205	8	8	2.34E-15
TP212	4	4	7.26E-19
TP242	11	11	2.88E-15
TP244	9	9	1.62E-14
TP267	26	25	1.95E-17
TP272	51	34	1.16E-10
TP307	12	12	124
TP327	9	9	.0285
TP333	9	9	.0433
TP334	8	8	.0538
TP350	24	24	.000245
TP358	5	5	.0000408
TP379	33	23	.0401
TREND	19	14	.0000386
TRIG_APP	7	7	.011
TUBTANK	4	4	.0118
VAPOR	9	9	.0000547
VISC_ELA	17	10	.0978
WEIBULL	9	9	.00101