IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. MARQ
8.4.1999
HBN/ms

DAMPING PARAMETER IN MARQUARDT'S METHOD

Hans Bruun Nielsen

TECHNICAL REPORT

IMM-REP-1999-05

IMM

DAMPING PARAMETER IN MARQUARDT'S METHOD

Hans Bruun Nielsen

Contents

1. Introduction

Given a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ (with $m \geq n$) and consider the problem of finding the least squares solution $\mathbf{x}^*$,

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x}} \left\{ F(\mathbf{x}) \equiv \tfrac{1}{2} \|\mathbf{f}(\mathbf{x})\|^2 \right\} \ , \tag{1.1}$$

where $\|\cdot\|$ denotes the Euclidean norm. When the components $f_i(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ are nonlinear functions, we have to use iteration: From a starting point $\mathbf{x}_0$ we compute $\mathbf{x}_1$, $\mathbf{x}_2$, ... and we shall assume that the descending condition

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k) \tag{1.2}$$

is satisfied. A general framework for such an algorithm is

$$
\begin{array}{l}
k := 0; \quad \mathbf{x} := \mathbf{x}_0 \\
\texttt{repeat} \\
\quad \text{Find a descent direction } \mathbf{h} \\
\quad \mathbf{x} := \mathbf{x} + \alpha\mathbf{h}; \qquad \{\text{with } \alpha > 0\} \\
\quad k := k{+}1 \\
\texttt{until}\ \textsc{Stop}
\end{array}
\tag{1.3}
$$

A descent direction satsifies

$$\mathbf{h}^\top \mathbf{F}'(\mathbf{x}) < 0 \ , \tag{1.4}$$

where the gradient $\mathbf{F}'$ is found as

$$\mathbf{F}'(\mathbf{x}) = \mathbf{J}_f(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) \tag{1.5a}$$

with the Jacobian matrix defined by

$$(\mathbf{J}_f(\mathbf{x}))_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}) \ . \tag{1.5b}$$

(When no confusion is possible we shall sometimes omit the argument $(\mathbf{x})$ of $\mathbf{f}$, $\mathbf{F}'$ and $\mathbf{J}_f$).

Relevant stopping criteria in algorithm (1.3) are

$$\|\mathbf{F}'(\mathbf{x})\| \le \varepsilon_1 , \quad \|\alpha \mathbf{h}\| \le \varepsilon_2 \|\mathbf{x}\| , \quad k \ge k_{\max} , \tag{1.6}$$

where ε_1, ε_2 and $k_{\max}$ are chosen by the user.

The simplest method in the framework (1.3) is based on using the *steepest descent direction*,

$$\mathbf{h} = -\mathbf{F}'(\mathbf{x}) ,$$

and computing α in (1.3) by line search. This method is robust when $\mathbf{x}$ is far from $\mathbf{x}^*$, but it has poor final convergence.

In *Newton's method* the step is found as the solution to

$$\mathbf{F}''(\mathbf{x})\mathbf{h} = -\mathbf{F}'(\mathbf{x}) , \tag{1.7a}$$

where

$$\mathbf{F}''(\mathbf{x}) = \mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x}) + \sum_{i=1}^{m} f_i(x)\mathbf{f}_i''(\mathbf{x}) . \tag{1.7b}$$

This method has quadratic final convergence (if $\mathbf{J}_f(\mathbf{x}^*)$ is non singular), but it is not robust. Also, it requires implementation of second derivatives, and it is rarely used in practice.

The *Gauss-Newton* method is based on the model obtained from the Taylor expansion of $\mathbf{f}$,

$$\begin{aligned} \mathbf{f}(\mathbf{x}+\mathbf{h}) &\simeq \boldsymbol{\ell}(\mathbf{h}) \equiv \mathbf{f}(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\mathbf{h} , \\ F(\mathbf{x}+\mathbf{h}) &\simeq L(\mathbf{h}) \equiv \tfrac{1}{2}\boldsymbol{\ell}(\mathbf{h})^\top \boldsymbol{\ell}(\mathbf{h}) . \end{aligned} \tag{1.8a}$$

The step is the minimizer of $L(\mathbf{h})$, which is the solution to

$$\left(\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})\right)\mathbf{h} = -\mathbf{F}'(\mathbf{x}) . \tag{1.8b}$$

The matrix $\mathbf{A} = \mathbf{J}_f^\top \mathbf{J}_f$ is symmetric and positive semidefinite. If $\mathbf{J}_f$ has full rank, then $\mathbf{A}$ is positive definite, implying that $\mathbf{h}$ satisfies (1.4). By comparison with (1.7) we see that if $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, then $\mathbf{A} \simeq \mathbf{F}''$ for $\mathbf{x}$ close to $\mathbf{x}^*$, and in this case the Gauss-Newton method also has quadratic final convergence. If the functions $\{f_i\}$ have small curvatures or if the $\{|f_i(\mathbf{x}^*)|\}$ are small, we can expect superlinear final convergence, but in general the final convergence is linear. It should be noted, however, that if the $\{\mathbf{f}_i''\}$ vary slowly with i, and if the $\{f_i\}$ behave like "white noise" as they do in many data fitting applications, then the "forgotten terms" in (1.7b) may almost cancel each other, so that it is possible also to get superlinear convergence in this case. This is illustrated in Figures 2.1 and 2.3 below.

This method shares the lack of robustness with Newton's method, and Levenberg [3] and later Marquardt [5] proposed to use what has become the most popular method, the *Marquardt algorithm* (also called the Levenberg–Marquardt algorithm). It can be outlined as follows,

$$\begin{array}{l} \mathbf{x} := \mathbf{x}_0; \quad \mu := \mu_0 \\ \texttt{repeat} \\ \quad \text{Solve } (\mathbf{J}_f^\top \mathbf{J}_f + \mu \mathbf{I})\mathbf{h} = -\mathbf{F}' \\ \quad \mathbf{x} := \mathbf{x} + \alpha \mathbf{h} \\ \quad \text{Update } \mu \\ \texttt{until } \textsc{Stop} \end{array} \tag{1.9}$$

Here, $\mathbf{I}$ is the identity matrix, and the *Marquardt (damping) parameter* μ is a strictly positive scalar. This implies that the matrix $\mathbf{J}_f^\top \mathbf{J}_f + \mu \mathbf{I}$ is positive definite, and $\mathbf{h}$ is in a descent direction. Therefore, $F(\mathbf{x}+\alpha\mathbf{h}) < F(\mathbf{x})$ for $\alpha > 0$ and sufficiently small.

If μ is small, then it follows that $\mathbf{h} \simeq \mathbf{h}_{\mathrm{GN}}$, the Gauss-Newton step defined by (1.8b), and if μ is large, then $\mathbf{h} \simeq -\frac{1}{\mu}\mathbf{F}'$, a short step in the steepest descent direction. Thus, the choice of μ affects both the direction and and the size of the step $\mathbf{h}$. If $\mathbf{x}$ is close to the solution, then we want the faster convergence of the Gauss-Newton method, while we prefer the robustness of the steepest descent method when $\mathbf{x}$ is far from $\mathbf{x}^*$.

The terms "big" and "small" must be related to the size of the elements in $\mathbf{J}_f^\top \mathbf{J}_f$. This matrix is symmetric and positive semidefinite, implying that the eigenvalues $\{\lambda_j\}$ are real and nonnegative, and the corresponding eigenvectors $\{\mathbf{v}_j\}$ can be chosen as an orthonormal basis of $\mathbb{R}^n$. The solution of the system defining the Marquardt step in (1.9) can be expressed in the form

$$\mathbf{h} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^\top \mathbf{F}'}{\lambda_j + \mu} \mathbf{v}_j \ .$$

Therefore, it is reasonable to relate the initial value μ_0 to the size of the eigenvalues. The maximum of the diagonal elements in the initial $\mathbf{J}_f^\top \mathbf{J}_f$ has the same order of magnitude as $\max\{\lambda_j\}$, so a simple strategy for choosing μ_0 is given by

$$\mu_0 = \tau \cdot \max\left\{ \left(\mathbf{J}_f(\mathbf{x}_0)^\top \mathbf{J}_f(\mathbf{x}_0) \right)_{ii} \right\} \ , \tag{1.10}$$

where a small value for τ is used if we believe that $\mathbf{x}_0$ is close to $\mathbf{x}^*$. In the test problems discussed in the next section we use τ-values in the range $[10^{-8}, 1]$.

In the next section we shall discuss some strategies for updating μ during the iteration and in Section 3 we give experimental results.

2. Updating the Damping Parameter

Basically there are two classes of strategies for updating the Marquardt damping parameter. One class is related to our formulation (1.9) of the algorithm: Find α by line search and use information from this to update μ. This is e.g. used in the MATLAB Optimization Toolbox implementation `leastsq`, [1].

A line search can involve many extra evaluations of $\mathbf{f}$, which do not get us closer to the solution, and the other class of strategies is based on the observation that through the choice of μ we influence both direction and size of $\mathbf{h}$, and can implement the method without a proper line search. If $F(\mathbf{x}+\mathbf{h}) < F(\mathbf{x})$, then $\mathbf{x}_{\text{new}} = \mathbf{x}+\mathbf{h}$, corresponding to $\alpha = 1$ in (1.9). Otherwise, $\mathbf{x}_{\text{new}} = \mathbf{x}$ (corresponding to $\alpha = 0$)

and μ is increased. With this strategy the second stopping criterion in (1.6) must be changed; we use

$$\|\mathbf{F}'(\mathbf{x})\| \le \varepsilon_1 \ , \tag{2.1a}$$

$$\|\mathbf{h}\| \le \varepsilon_2 \|\mathbf{x}\| \ , \tag{2.1b}$$

$$k \ge k_{\max} \ . \tag{2.1c}$$

We also want to be able to decrease μ when $\mathbf{x}$ gets close to $\mathbf{x}^*$. This is decided by the step being so small that $L(\mathbf{h})$ is a good approximation to $F(\mathbf{x}+\mathbf{h})$. This can be expressed in terms of the *"gain factor"*

$$\varrho = \frac{F(\mathbf{x}) - F(\mathbf{x}+\mathbf{h})}{L(\mathbf{0}) - L(\mathbf{h})} \ . \tag{2.2}$$

From (1.8) and (1.9) we see that

$$\begin{aligned} L(\mathbf{0}) - L(\mathbf{h}) &= -\mathbf{h}^\top \mathbf{J}_f^\top \mathbf{f} - \tfrac{1}{2}\mathbf{h}^\top \mathbf{J}_f^\top \mathbf{J}_f \mathbf{h} \\ &= -\tfrac{1}{2}\mathbf{h}^\top [2\mathbf{F}' + (\mathbf{J}_f^\top \mathbf{J}_f + \mu\mathbf{I} - \mu\mathbf{I})\mathbf{h}] \\ &= \tfrac{1}{2}\mathbf{h}^\top (\mu\mathbf{h} - \mathbf{F}') \ . \end{aligned} \tag{2.3}$$

Since both $\mu\mathbf{h}^\top\mathbf{h}$ and $-\mathbf{h}^\top\mathbf{F}'$ are positive, it follows that the nominator in (2.2) is positive, and $\varrho > 0$ is equivalent with the descending condition (1.2) being satisfied with $\mathbf{x}_k = \mathbf{x}$, $\mathbf{x}_{k+1} = \mathbf{x}+\mathbf{h}$. If ϱ is large, then we can decrease μ in order to get closer to the Gauss-Newton direction in the next iteration step.

If $\varrho \le 0$, then we do not change $\mathbf{x}$, but increase μ with the twofold purpose of getting closer to the steepest descent direction and reduce the step length. Also if $\varrho > 0$ but small, it might be better to use a larger damping parameter in the next iteration step.

Most implementations with this strategy have the following form, basically as proposed by Marquardt [5],

$$\begin{array}{ll} \texttt{if} \ \varrho < \varrho_1 \ \texttt{then} \ \mu := \beta * \mu \\ \texttt{if} \ \varrho > \varrho_2 \ \texttt{then} \ \mu := \mu/\gamma \\ \texttt{if} \ \varrho > 0 \ \texttt{then} \ \mathbf{x} := \mathbf{x} + \mathbf{h} \end{array} \tag{2.4}$$

where $0 < \varrho_1 < \varrho_2 < 1$ and $\beta, \gamma > 1$. In the next section we give results from experiments with these parameters. The outcome is that the method is not very sensitive to small changes in the values, and the popular choice $\varrho_1 = 0.25$, $\varrho_2 = 0.75$, $\beta = 2$ and $\gamma = 3$ is good in many cases.

Figure 2.1 illustrates the performance of this strategy on a data fitting problem (problem 18 in the appendix) with $n = 4$, $m = 45$. We used $\tau_0 = 10^{-3}$ in (1.10) and $\varepsilon_1 = \varepsilon_2 = 10^{-10}$, $k_{\max} = 100$ in the stopping criteria (2.1).

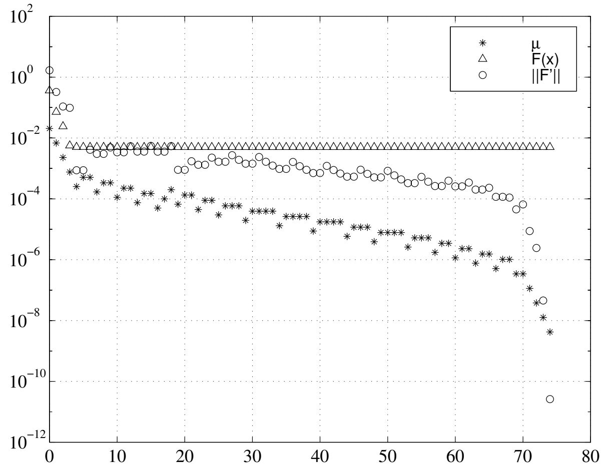

Figure 2.1. *Marquardt's method with updating strategy (2.4)*

The iteration was stopped by (2.1a) after 74 steps. From step 20 to step 68 we see that each decrease in μ is immediately followed by an increase, and the norm of the gradient has a rugged behaviour. The final convergence seems to be superlinear, cf. the discussion on p. 6.

We propose a new updating strategy that has the same simplicity as (2.4) but avoids the jumps in μ_{new}/μ across the thresholds ϱ_1 and ϱ_2. Further, if $\varrho < 0$ in consecutive steps, then we let μ grow faster,

$$\begin{array}{l} \texttt{if} \ \varrho > 0 \ \texttt{then} \\ \quad \mathbf{x} := \mathbf{x} + \mathbf{h} \\ \quad \mu := \mu * \max\{1/\gamma, \ 1 - (\beta-1)(2\varrho-1)^p\}\,; \quad \nu := \beta \\ \texttt{else} \ \mu := \mu * \nu; \quad \nu := 2 * \nu \end{array} \tag{2.5}$$

with ν initialized to β and p being an odd integer. This is illustrated in Figure 2.2. For comparison we also indicate the discontinuous updating by (2.4).

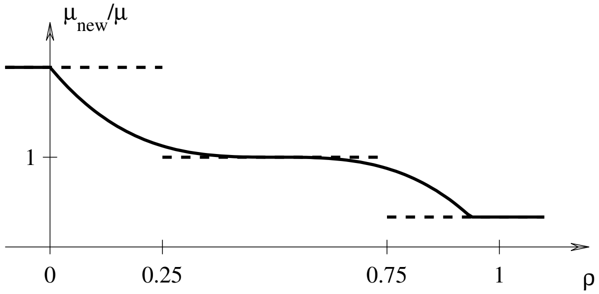

Figure 2.2. *Updating strategies (2.5) (full line) and (2.4) (dashed line)*
$\beta = \nu = 2, \ \gamma = 3, \ p = 3$

In Figure 2.3 we show the performance with this strategy on the same problem as in Figure 2.1. It is seen that the smoother updating rusults in a smoother performance and faster convergence: the number of iteration steps is reduced from 74 to 62.

In the next section we verify experimentally, that this example is not uniqe, but in most cases the new updating strategy does indeed perform better.

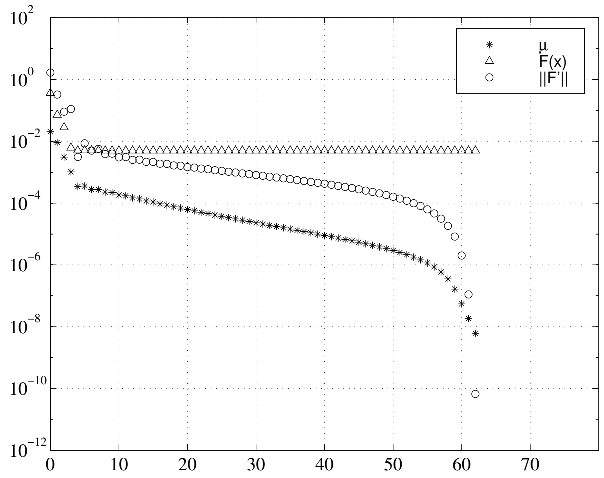

Figure 2.3. *Marquardt's method with updating strategy (2.4)*

3. Numerical Experiments

The best choice of the parameters in the updating formulae (2.4) and (2.5) depends on the problem and the starting point (both $\mathbf{x}_0$ and μ_0). To get an idea of what values give an overall good performance, we experimented with 20 different problems, specified in the appendix. With some of them we could vary the value of n and m, giving a total of 30 problems.

The computation was performed in MATLAB with an accuracy of about 17 decimal digits. In all cases we used the stopping criteria (2.1) with $\varepsilon_2 = 10^{-12}$ and $k_{\max} = 500$.

First, consider the updating strategy (2.4). In Table 3.1 we give the number of iteration steps for a number of choices of the parameters. It should by noted that we simplified the testing by letting $\varrho_2 = 1 - \varrho_1$.

		β	2	2	3	3	2	2
		γ	3	2	3	2	3	3
		$\varrho_1 = 1-\varrho_2$	.25	.25	.25	.25	.20	.15
Pno	m	n						
1	8	8	1	1	1	1	1	1
	32	16	1	1	1	1	1	1
2	8	8	2	2	2	2	2	2
	32	16	2	2	2	2	2	2
3	8	8	2	2	2	2	2	2
	32	16	2	2	2	2	2	2
4	2	2	27	26	24	28	24	24
5	3	3	12	15	12	15	12	14
6	4	4	10	10	10	10	10	10
7	2	2	58	61	39	39	45	52
8	15	3	5	5	5	5	5	5
9	11	4	16	19	16	19	16	16
10	16	3	231	311	422	225	214	235
11	31	6	6	6	6	6	6	6
	31	9	4	4	4	4	4	4
	31	12	4	4	4	4	4	4
12	5	3	4	4	4	4	4	4
	10	3	5	5	5	5	5	5
13	10	2	18	36	17	32	18	18
14	20	4	65	68	55	53	63	73
15	8	8	33	27	28	31	33	32
	16	8	72	59	57	50	73	73
	9	9	10	11	10	11	10	10
	18	9	24	25	24	25	24	24
16	5	5	9	13	9	13	9	9
	10	10	11	15	11	15	11	11
17	33	5	17	21	24	22	17	17
18	45	4	73	82	121	83	76	84
19	45	2	19	22	15	18	19	19
20	16	3	111	116	166	112	91	92
		Sum :	854	975	1098	839	803	851
		Accum. dev :	104	225	348	89	53	101

Table 3.1a. *Strategy (2.4)*, $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-12}$

		β	2	2	3	3	2	2
		γ	3	2	3	2	3	3
		$\varrho_1 = 1 - \varrho_2$	.25	.25	.25	.25	.20	.15
Pno	m	n						
1	8	8	2	2	2	2	2	2
	32	16	3	3	3	3	3	3
2	8	8	4	2	4	2	4	4
	32	16	12	7	9	6	12	12
3	8	8	2	2	2	2	2	2
	32	16	14	13	8	7	14	14
4	2	2	29	29	26	30	27	27
5	3	3	14	18	14	18	14	16
6	4	4	16	16	16	16	16	16
7	2	2	58	61	51	48	62	52
8	15	3	7	7	7	7	7	7
9	11	4	49	44	33	47	49	49
10	16	3	231	311	422	225	214	235
11	31	6	52	20	37	23	50	50
	31	9	11	12	11	12	10	10
	31	12	16	22	16	22	17	17
12	5	3	5	5	5	5	5	5
	10	3	6	6	6	6	6	6
13	10	2	28	36	28	32	28	28
14	20	4	65	68	55	53	63	73
15	8	8	61	48	47	46	57	57
	16	8	92	76	72	65	95	100
	9	9	12	15	12	15	12	12
	18	9	89	71	66	58	89	89
16	5	5	12	16	12	16	12	12
	10	10	13	18	13	18	13	13
17	33	5	30	40	29	31	30	30
18	45	4	106	84	141	91	84	87
19	45	2	43	30	40	24	43	43
20	16	3	127	137	178	119	96	95
		Sum :	1209	1219	1365	1049	1136	1166
		Accum. dev :	253	263	409	93	180	210

Table 3.1b. *Strategy (2.4)*, $\varepsilon_1 = \varepsilon_2 = 10^{-12}$

Problems 5 and 10 are noted for their difficulty, [4]. As can be seen from Tables 3.1a-b we do not find Problem 5 to be specially hard, whereas we agree on Problem 10. The difficulty is partly caused by the components of $\mathbf{x}^*$ having markedly different orders of magnitude, and so have the columns of the Jacobian matrix. This is remedied in Problem 20, resulting in halving the number of iteration steps.

None of the six choices of parameters comes out as a winner in all 30 cases. In an attempt to clarify matters we summarize the tables by giving *Sum*: the number of iteration steps accumulated over the 30 tests and *Accum. dev* $= \sum_j (it_j - \mathcal{M}_j)$, where $\mathcal{M}_j$ is the minimum over the 6 choices of parameters, e.g. in Table 3.1a, $\mathcal{M}_{29} = 15$ and $\mathcal{M}_{30} = 91$.

Even with this attempt at simplifying matters we do not get a clear winner, but the choice $\beta = 2$, $\gamma = 3$, seems to be good, and $(\varrho_1, \varrho_2) = (0.20, 0.80)$ seems to be slightly better than the popular $(\varrho_1, \varrho_2) = (0.25, 0.75)$.

In Table 3.2 we give results from similar experiments with the updating strategy (2.5). The number of iteration steps do not vary so much, and further investigation shows that this is mainly because of the "ν-updating": When the algorithm cannot improve the current iterate, then (2.5) quickly increases μ, and the process is stopped by (2.1b). This is further illustrated in Table 3.3 and Figure 3.1. The choice $\beta = 2$, $\gamma = 3$, $p = 3$ seems to be good.

		β	2	2	3	2	2	2
		γ	3	3	2	2.5	4	5
		p	3	5	3	3	3	3
Pno	m	n						
1	8	8	1	1	1	1	1	1
	32	16	1	1	1	1	1	1
2	8	8	2	2	2	2	2	2
	32	16	2	2	2	2	2	2
3	8	8	2	2	2	2	2	2
	32	16	2	2	2	2	2	2
4	2	2	23	26	27	26	23	23
5	3	3	14	14	16	15	12	12
6	4	4	10	10	10	10	10	10
7	2	2	36	51	41	42	39	36
8	15	3	5	5	5	5	5	5
9	11	4	16	16	18	15	15	15
10	16	3	181	198	185	184	170	176
11	31	6	6	6	6	6	6	6
	31	9	4	4	4	4	4	4
	31	12	4	4	4	4	4	4
12	5	3	4	4	4	4	4	4
	10	3	5	5	5	5	5	5
13	10	2	27	31	22	24	17	17
14	20	4	42	42	44	43	42	42
15	8	8	31	31	31	28	30	26
	16	8	41	41	41	40	41	44
	9	9	10	10	11	10	8	9
	18	9	24	24	25	24	23	23
16	5	5	9	9	13	11	8	8
	10	10	11	11	15	12	9	9
17	33	5	15	15	20	17	12	13
18	45	4	61	70	60	61	64	65
19	45	2	12	13	11	13	11	11
20	16	3	88	88	93	89	83	83
		Sum :	689	738	721	702	655	660
		Accum. dev :	46	95	78	59	12	17

Table 3.2a. *Strategy (2.5)*, $\varepsilon_1 = = 10^{-6}$, $\varepsilon_2 = 10^{-12}$

		β	2	2	3	2	2	2
		γ	3	3	2	2.5	4	5
		p	3	5	3	3	3	3
Pno	m	n						
1	8	8	2	2	2	2	2	2
	32	16	3	3	3	3	4	4
2	8	8	4	4	2	3	11	2
	32	16	7	7	5	7	13	11
3	8	8	2	2	2	2	2	2
	32	16	10	10	6	7	7	8
4	2	2	25	28	30	29	25	24
5	3	3	16	17	18	17	14	14
6	4	4	16	16	16	16	16	16
7	2	2	42	51	41	42	39	45
8	15	3	7	7	7	8	7	7
9	11	4	39	33	35	36	41	39
10	16	3	181	198	185	184	170	176
11	31	6	19	22	21	19	21	25
	31	9	9	9	12	10	9	15
	31	12	16	16	23	19	14	13
12	5	3	5	5	5	5	5	5
	10	3	6	6	6	6	6	6
13	10	2	27	31	22	24	17	17
14	20	4	42	42	44	43	42	42
15	8	8	50	49	44	46	42	39
	16	8	60	66	50	60	59	55
	9	9	12	12	15	13	11	11
	18	9	47	44	39	52	46	44
16	5	5	12	12	16	14	10	10
	10	10	13	13	18	15	11	10
17	33	5	22	24	31	23	18	20
18	45	4	66	80	63	67	79	72
19	45	2	24	19	34	25	22	23
20	16	3	96	98	100	95	89	90
		Sum :	880	926	895	892	852	847
		Accum. dev :	98	144	113	110	70	65

Table 3.2b. *Strategy (2.5)*, $\varepsilon_1 = \varepsilon_2 = 10^{-12}$

In table 3.3 we compare results from the two strategies with the "best" choice of parameters. *Jev* denotes the number of evaluations of $\mathbf{f}(\mathbf{x})$ and $\mathbf{J}_f(\mathbf{x})$, and we also give the norm of the gradient at the final iterate. Except when $F(\mathbf{x}^*) = 0$, the two strategies give identical values for F at the final iterate (with the three digits shown), and there does not seem to be a systematic difference between the accuracy obtained, as measured by $\|\mathbf{F}'\|$. In three cases strategy (2.4) is faster than (2.5): Problem 5 with both choices of desired accuracy and Problem 13 in Table 3.3a. In some of the other cases the two strategies perform (almost) equally, but the accumulated numbers show that with the crude accuracy the updating strategy (2.5) is 17% faster than (2.4), and with the fine accuracy the difference is 29%.

The difference is partly caused by the "ν-updating". This is illustrated in Figure 3.1, and commented after Tables 3.3a–b.

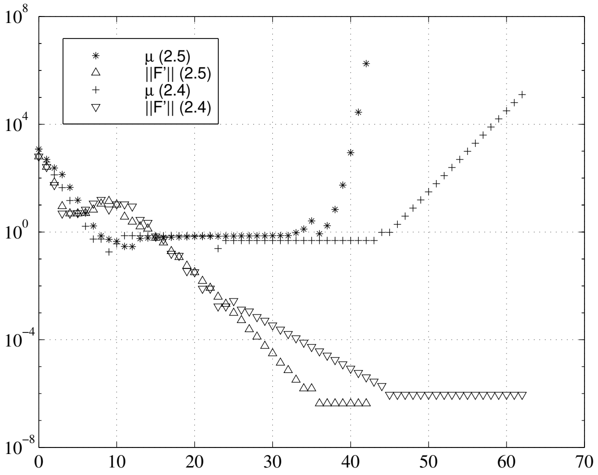

Figure 3.1. *Problem 7 with* $\varepsilon_1 = \varepsilon_2 = 10^{-12}$

			Strategy	(2.4)		(2.5)	
Pno	m	n	$F(\mathbf{x}^*)$	Jev	$\|\mathbf{F}'\|$	Jev	$\|\mathbf{F}'\|$
1	8	8	1.60e-15	2	5.66e-08	2	5.66e-08
	32	16	8.00e+00	2	8.00e-08	2	8.00e-08
2	8	8	8.24e-01	3	1.28e-12	3	1.28e-12
	32	16	3.82e+00	3	3.67e-10	3	3.67e-10
3	8	8	1.58e+00	3	7.35e-14	3	7.35e-14
	32	16	4.57e+00	3	3.04e-10	3	3.04e-10
4	2	2	4.09e-13	25	9.38e-07	24	1.46e-08
5	3	3	8.58e-15	13	1.11e-07	15	8.03e-09
6	4	4	7.32e-11	11	2.11e-07	11	2.11e-07
7	2	2	2.45e+01	46	9.04e-07	37	4.29e-07
8	15	3	4.11e-03	6	3.04e-09	6	3.04e-09
9	11	4	1.54e-04	17	7.11e-07	17	7.27e-07
10	16	3	4.40e+01	215	8.09e-04	182	4.19e-04
11	31	6	1.14e-03	7	8.08e-07	7	8.09e-07
	31	9	1.15e-06	5	4.56e-07	5	4.25e-07
	31	12	1.81e-08	5	2.14e-07	5	2.11e-07
12	5	3	4.91e-15	5	1.17e-07	5	1.17e-07
	10	3	5.69e-20	6	4.88e-10	6	4.88e-10
13	10	2	6.22e+01	19	1.89e-07	28	1.12e-06
14	20	4	4.29e+04	64	9.26e-05	43	1.14e-04
15	8	8	1.76e-03	34	8.14e-07	32	5.12e-07
	16	8	2.95e-02	74	9.67e-07	42	6.41e-07
	9	9	1.07e-15	11	2.43e-08	11	1.45e-08
	18	9	3.55e-02	25	6.20e-07	25	6.77e-07
16	5	5	5.01e-12	10	6.04e-07	10	6.04e-07
	10	10	1.99e-14	12	2.20e-08	12	2.20e-08
17	33	5	2.73e-05	18	3.26e-08	16	1.75e-08
18	45	4	5.00e-03	77	9.27e-08	62	1.10e-07
19	45	2	5.00e-03	20	1.04e-07	13	1.40e-07
20	16	3	4.40e-05	92	4.00e-09	89	2.71e-09
			Sum :	833		719	

Table 3.3a. *Crude accuracy,* $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-12}$.
Strategy (2.4) with $(\beta, \gamma, \varrho_1, \varrho_2) = (2, 3, 0.2, 0.8)$
Strategy (2.5) with $(\beta, \gamma, p) = (2, 3, 3)$

			Strategy	(2.4)		(2.5)	
Pno	m	n	$F(\mathbf{x}^*)$	Jev	$\Vert\mathbf{F}'\Vert$	Jev	$\Vert\mathbf{F}'\Vert$
1	8	8	0	3	0	3	0
	32	16	8.00e+00	4	8.64e-16	4	8.64e-16
2	8	8	8.24e-01	5	8.52e-13	5	8.52e-13
	32	16	3.82e+00	13	1.85e-10	8	1.85e-10
3	8	8	1.58e+00	3	7.35e-14	3	7.35e-14
	32	16	4.57e+00	15	9.10e-11	11	9.10e-11
4	2	2	7.12e-30	28	3.77e-15	26	2.92e-14
5	3	3	1.50e-26	15	1.47e-13	17	5.09e-15
6	4	4	4.36e-18	17	8.04e-13	17	8.04e-13
7	2	2	2.45e+01	63	9.04e-07	43	4.29e-07
8	15	3	4.11e-03	8	4.84e-13	8	4.84e-13
9	11	4	1.54e-04	50	5.66e-12	40	3.49e-12
10	16	3	4.40e+01	215	8.09e-04	182	4.19e-04
11	31	6	1.14e-03	51	2.67e-11	20	2.11e-10
	31	9	7.00e-07	11	6.48e-14	10	4.68e-13
	31	12	2.36e-10	18	1.36e-13	17	7.09e-13
12	5	3	8.05e-28	6	3.41e-14	6	3.41e-14
	10	3	1.93e-33	7	9.47e-17	7	9.47e-17
13	10	2	6.22e+01	29	1.89e-07	28	1.12e-06
14	20	4	4.29e+04	64	9.26e-05	43	1.14e-04
15	8	8	1.76e-03	58	2.11e-09	51	1.37e-09
	16	8	2.95e-02	96	3.14e-09	61	3.32e-08
	9	9	2.54e-26	13	1.18e-13	13	5.52e-14
	18	9	3.55e-02	90	5.73e-09	48	1.80e-08
16	5	5	2.09e-26	13	3.52e-14	13	3.52e-14
	10	10	9.17e-24	14	3.93e-13	14	3.93e-13
17	33	5	2.73e-05	31	6.03e-13	23	4.22e-13
18	45	4	5.00e-03	85	4.37e-13	67	1.82e-13
19	45	2	5.00e-03	44	3.23e-13	25	2.87e-12
20	16	3	4.40e-05	97	1.63e-12	97	1.39e-11
			Sum :	1166		910	

Table 3.3b. *Fine accuracy,* $\varepsilon_1 = \varepsilon_2 = 10^{-12}$.
Strategy (2.4) with $(\beta, \gamma, \varrho_1, \varrho_2) = (2, 3, 0.2, 0.8)$
Strategy (2.5) with $(\beta, \gamma, p) = (2, 3, 3)$

Around iteration step no. 10 both versions of Marquardt's method have come reasonably close to the solution, and as in Figures 2.1 and 2.3 the smoother updating (2.5) is seen to work best: μ is increased slightly in each step and the (linear) convergence is faster. The limit of the method is reached at step no. 36 with strategy (2.5), step no. 45 with strategy (2.4). The stopping criterion (2.1a) is not satisfied, and it takes 6 more steps with (2.5) before the iteration is stopped by (2.1b); 17 steps with strategy (2.4).

Finally, in Table 3.4 we give results from the MATLAB version 5.2 Optimization Toolbox implementation `leastsq`. The two choices of accuracy were defined by setting `op(2)` $=$ `op(3)` $= \varepsilon$, where `op` is the Option Parameter vector, see p. 1-25 in [1]. The accumulated number of Marquardt steps (as given by *Jev*) is seen to be about the same as we use with strategy (2.4), but on top of that the line searches require more than twice that number of evaluations of $\mathbf{f}(\mathbf{x})$ alone, denoted *fev* in the table.

A comparison with strategy (2.5) in the fine accuracy case, $\varepsilon_1 = \varepsilon = 10^{-12}$ shows that `leastsq` needs less Marquardt steps with 11 out of the 30 problems. However, for $(Pno, m, n) = (2, 32, 16)$ and $(3, 32, 16)$ the obtained accuracy, as measured by $\Vert\mathbf{F}'\Vert$, from `leastsq` is inferior by a factor larger than 10^3. When we ignore *fev*, the best results from `leastsq` are for $(Pno, m, n) = (15, 16, 8), (15, 18, 9), (19, 45, 2)$, where the values for $\Vert\mathbf{F}'\Vert$ are of the same order of magnitude as obtained by strategy (2.5), and the number of Marquardt steps is about halved.

The implementation of `leastsq` is not robust: For $(Pno, \varepsilon) = (6, 10^{-12})$ it enters an infinite loop. At the solution $\mathbf{x}^* = \mathbf{0}$ the Jacobian matrix has rank 2 (while $n = 4$). A closer investigation reveals that after 45 Marquardt steps the current iterate is close to the solution: $\Vert\mathbf{x}\Vert \simeq 4 \cdot 10^{-9}$ and $\mu \simeq 2.25 \cdot 10^{-16}$. This is so small that the matrix $\mathbf{A} = \mathbf{J}_f^\top \mathbf{J}_f + \mu\mathbf{I}$ is singular to working precision, resulting in $\mathbf{h} = \infty$, and the ensuing line search never stops! The algorithm has a stopping criterion like (2.1c), but this is outside the line search loop.

Accuracy			$\varepsilon = 10^{-6}$			$\varepsilon = 10^{-12}$		
Pno	m	n	Jev	fev	$\|\mathbf{F}'\|$	Jev	fev	$\|\mathbf{F}'\|$
1	8	8	5	11	1.09e-09	6	13	3.14e-16
	32	16	4	8	3.41e-06	7	16	1.20e-09
2	8	8	5	9	4.26e-09	6	11	6.23e-13
	32	16	5	9	9.07e-06	7	14	9.07e-06
3	8	8	5	9	8.20e-10	6	11	1.14e-12
	32	16	6	11	1.61e-07	7	13	1.61e-07
4	2	2	17	36	1.63e-07	19	40	3.65e-12
5	3	3	15	34	1.20e-07	17	38	1.47e-12
6	4	4	31	89	5.84e-11	*	*	*
7	2	2	32	79	4.91e-06	39	94	3.53e-08
8	15	3	12	27	3.62e-07	16	39	2.60e-09
9	11	4	11	21	1.75e-09	15	31	4.33e-11
10	16	3	398	925	1.17e-03	398	925	1.17e-03
11	31	6	19	46	9.97e-09	22	53	1.32e-10
	31	9	25	65	2.43e-13	27	72	6.55e-13
	31	12	37	100	4.47e-13	41	108	4.19e-13
12	5	3	18	46	4.28e-10	20	50	9.08e-15
	10	3	19	46	4.56e-09	22	52	3.74e-16
13	10	2	9	18	1.77e-06	12	23	1.53e-07
14	20	4	49	125	1.40e-04	49	125	1.40e-04
15	8	8	106	210	2.26e-06	162	330	2.51e-08
	16	8	19	40	1.53e-05	27	65	8.01e-08
	9	9	9	18	9.63e-08	12	24	4.32e-14
	18	9	15	28	7.78e-06	26	52	1.37e-08
16	5	5	7	14	8.34e-08	9	18	1.52e-12
	10	10	9	18	1.40e-07	12	24	9.13e-15
17	33	5	22	51	2.00e-07	25	59	1.12e-10
18	45	4	70	162	1.30e-08	71	167	1.30e-08
19	45	2	11	32	2.27e-10	14	47	8.21e-11
20	16	3	103	220	1.49e-06	104	234	1.49e-06
Sum :			1093	2507		1198	2748	

Table 3.4. `leastsq` *from* MATLAB *Optimization Toolbox*
*An * indicates an infinite loop*

4. Conclusion

Marquardt's method seems to be most efficiently implemented without a proper line search, and the proposed smooth variation of the damping parameter with accelerated increase in case the algorithm gets stuck is significantly better than the classical strategy proposed by Marquardt.

We have investigated only least squares problems, but the idea is used also in other nonlinear optimization algorithms, see e.g. [2]. We believe that the new updating will also be advantageous in such applications.

Appendix. Test Problems

We use a set of test problems, where the first 17 were taken from [4]. They originate from Argonne National Laboratory, USA. Most of these problems have appeared in the optimization literature, and we give the names which are usually connected with these functions.

In the list below we use the following format

a) Dimension
b) Function definition
c) Starting point $\mathbf{x}_0$ and τ_0
d) Solution

In the presentation $\mathbf{e}$ ($\mathbf{E}$) denotes the vector (matrix) of all ones, and $\mathbf{I}$ is the identity matrix.

1. *Linear function, full rank*

a) n variable, $m \ge n$

b) $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{e}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{I} - \frac{2}{m}\mathbf{E} \\ -\frac{2}{m}\mathbf{E} \end{bmatrix}$

c) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}$

d) $\mathbf{x}^* = -\mathbf{e}, \quad F(\mathbf{x}^*) = \frac{1}{2}(m-n)$

2. *Linear function, rank 1*

a) n variable, $m \ge n$

b) $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{e}, \quad \mathbf{A} = \begin{bmatrix} 1 \\ \vdots \\ m \end{bmatrix} \begin{bmatrix} 1 & \cdots & n \end{bmatrix}$

c) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}$

d) Any $\mathbf{x}^*$ such that $\begin{bmatrix} 1 & \cdots & n \end{bmatrix} \mathbf{x}^* = \dfrac{3}{2m+1}, \quad F(\mathbf{x}^*) = \dfrac{m(m-1)}{4(2m+1)}$

3. *Linear function, rank 1 with zero columns and rows*

a) n variable, $n \ge 3, \quad m \ge n$

b) $\mathbf{f}(\mathbf{x}) = \widetilde{\mathbf{A}}\mathbf{x} - \mathbf{e}, \quad \widetilde{\mathbf{A}} = \left[\begin{array}{c|c|c} 0 & \mathbf{0}^\top & 0 \\ \hline \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \hline 0 & \mathbf{0}^\top & 0 \end{array}\right],$

$\mathbf{A} = \begin{bmatrix} 1 \\ \vdots \\ m-2 \end{bmatrix} \begin{bmatrix} 2 & \cdots & n-2 \end{bmatrix}$

c) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}$

d) Any $\mathbf{x}^*$ such that $\begin{bmatrix} 0 & 2 & \cdots & n-2 & 0 \end{bmatrix} \mathbf{x}^* = \dfrac{3}{2m-3},$

$F(\mathbf{x}^*) = \dfrac{m^2 + 3m - 6}{4(2m-3)}$

4. *Rosenbrock function*

a) $n = m = 2$

b) $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 10(x_2 - x_1^2) \\ 1 - x_1 \end{bmatrix}$

c) $\mathbf{x}_0 = [\,-1.2,\ 1\,]^\top, \quad \tau_0 = 1$

d) $\mathbf{x}^* = \mathbf{e}, \quad F(\mathbf{x}^*) = 0$

5. *Helical valley function*

a) $n = m = 3$

b) $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 10\left(x_3 - \frac{5}{\pi}\arctan\frac{x_2}{x_1} - \theta\right) \\ 10\left(\sqrt{x_1^2 + x_2^2} - 1\right) \\ x_3 \end{bmatrix} \quad \text{with } \theta = \begin{cases} 0 & x_1 > 0 \\ 5 & x_1 < 0 \end{cases}$

c) $\mathbf{x}_0 = [-1\ \ 0\ \ 0]^\top, \quad \tau_0 = 1$

d) $\mathbf{x}^* = [1\ \ 0\ \ 0]^\top, \quad F(\mathbf{x}^*) = 0$

6. *Powell singular function*

a) $n = m = 4$

b) $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1 + 10x_2 \\ \sqrt{5}(x_3 - x_4) \\ (x_2 - 2x_3)^2 \\ \sqrt{10}(x_1 - x_4)^2 \end{bmatrix}$

c) $\mathbf{x}_0 = [3 \quad -1 \quad 0 \quad 1]^\top, \quad \tau_0 = 10^{-8}$

d) $\mathbf{x}^* = \mathbf{0}, \quad F(\mathbf{x}^*) = 0$

7. *Freudenstein and Roth function*

a) $n = m = 2$

b) $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1 - x_2 * (2 - x_2 * (5 - x_2)) - 13 \\ x_1 - x_2 * (14 - x_2 * (1 + x_2)) - 29 \end{bmatrix}$

c) $\mathbf{x}_0 = [0.5, \ -2]^\top, \quad \tau_0 = 1$

d) $\mathbf{x}^* \simeq [11.4128, \ -0.896805]^\top, \quad F(\mathbf{x}^*) \simeq 24.4921$

8. *Bard function*

a) $n = 3, \quad m = 15$

b) $f_i(\mathbf{x}) = y_i - \left(x_1 + \dfrac{u_i}{x_2 v_i + x_3 w_i}\right)$

c) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 0.082411 & 1.133036 & 2.343695 \end{bmatrix}^\top, \quad F(\mathbf{x}^*) \simeq 4.10744 \cdot 10^{-3}$

Data: $u_i = i, \ v_i = 16 - i, \ w_i = \min\{u_i, v_i\}$ and

i	y_i	i	y_i	i	y_i
1	0.14	6	0.32	11	0.73
2	0.18	7	0.35	12	0.96
3	0.22	8	0.39	13	1.34
4	0.25	9	0.37	14	2.10
5	0.29	10	0.58	15	4.39

9. *Kowalik and Osborne function*

a) $n = 4, \quad m = 11$

b) $f_i(\mathbf{x}) = y_i - \dfrac{x_1 u_i (u_i + x_2)}{u_i (u_i + x_3) + x_4}$

c) $\mathbf{x}_0 = \begin{bmatrix} 0.25 & 0.39 & 0.415 & 0.39 \end{bmatrix}^\top, \quad \tau_0 = 1$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 0.192807 & 0.191282 & 0.123057 & 0.136062 \end{bmatrix}^\top,$
$F(\mathbf{x}^*) \simeq 1.53753 \cdot 10^{-4}$

Data:

i	y_i	u_i	i	y_i	u_i
1	0.1957	4.0000	7	0.0456	0.1250
2	0.1947	2.0000	8	0.0342	0.1000
3	0.1735	1.0000	9	0.0323	0.0833
4	0.1600	0.5000	10	0.0235	0.0714
5	0.0844	0.2500	11	0.0246	0.0625
6	0.0627	0.1670			

10. *Meyer function*

a) $n = 3, \quad m = 16$

b) $f_i(\mathbf{x}) = x_1 \ \exp\left(\dfrac{x_2}{t_i + x_3}\right) - y_i$

c) $\mathbf{x}_0 = \begin{bmatrix} 0.02 & 4000 & 250 \end{bmatrix}^\top, \quad \tau_0 = 1$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 0.00560964 & 6181.35 & 345.224 \end{bmatrix}^\top, \quad F(\mathbf{x}^*) \simeq 43.9729$

Data: $t_i = 45 + 5i$,

i	y_i	i	y_i	i	y_i
1	34780	7	11540	12	5147
2	28610	8	9744	13	4427
3	23650	9	8261	14	3820
4	19630	10	7030	15	3307
5	16370	11	6005	16	2872
6	13720				

11. *Watson function*

a) $2 \le n \le 31, \quad m = 31$

b) $$f_i(\mathbf{x}) = \begin{cases} \displaystyle\sum_{j=2}^{n} (j-1) t_i^{j-2} x_j - \left(\sum_{j=1}^{n} t_i^{j-1} x_j\right)^2 - 1 & i = 1, \ldots, 29 \\ x_1 & i = 30 \\ x_2 - x_1^2 - 1 & i = 31 \end{cases}$$

where $t_i = i/29$.

c) $\mathbf{x}_0 = \mathbf{0}, \quad \tau_0 = 10^{-8}$

d)

n	6	9	12
$F(\mathbf{x}^*) \simeq$	$1.143835 \cdot 10^{-3}$	$6.998801 \cdot 10^{-7}$	$2.361196 \cdot 10^{-10}$

12. Box 3-dimensional function

a) $n=3, \quad m \geq 3$ variable

b) $f_i(\mathbf{x}) = e^{-x_1 t_i} - e^{-x_2 t_i} - x_3 \left(e^{-t_i} - e^{-10 t_i}\right)$
where $t_i = i/10$

c) $\mathbf{x}_0 = \begin{bmatrix} 0 & 10 & 20 \end{bmatrix}^\top, \quad \tau_0 = 10^{-8}$

d) $\mathbf{x}^* = \begin{bmatrix} 1 & 10 & 1 \end{bmatrix}^\top, \ \mathbf{x}^* = \begin{bmatrix} 10 & 1 & -1 \end{bmatrix}^\top$ or $\mathbf{x}^* = \begin{bmatrix} \alpha & \alpha & 0 \end{bmatrix}^\top$
for any $\alpha \in \mathbb{R}$. $F(\mathbf{x}^*) = 0$

13. Jennrich and Sampson function

a) $n=2, \quad m \geq 2$ variable

b) $f_i(\mathbf{x}) = 2+2i - \left(e^{x_1 i} + e^{x_2 i}\right)$

c) $\mathbf{x}_0 = [0.3,\ 0.4]^\top, \quad \tau_0 = 1$

d) $m=5 \ : \ x_1^* = x_2^* \simeq 0.378468, F(\mathbf{x}^*) \simeq 4.8879031$
$m=10 : \ x_1^* = x_2^* \simeq 0.257825, F(\mathbf{x}^*) \simeq 62.1811$
$m=20 : \ x_1^* = x_2^* \simeq 0.165191, F(\mathbf{x}^*) \simeq 724.740$

14. Brown and Dennis function

a) $n=4, \quad m \geq 4$ variable

b) $f_i(\mathbf{x}) = \left(x_1 + x_2 t_i - e^{t_i}\right)^2 + (x_3 + x_4 \sin t_i - \cos t_i)^2$

c) $\mathbf{x}_0 = \begin{bmatrix} 25 & 5 & -5 & -1 \end{bmatrix}^\top, \quad \tau_0 = 10^{-8}$

d) $m=5 \quad : \ \mathbf{x}^* \simeq \begin{bmatrix} 0.77781 & 1.87187 & 1.15301 & -0.67095 \end{bmatrix}^\top,$
$F(\mathbf{x}^*) \simeq 9.08309 \cdot 10^{-5}$
$m=10 \quad : \ \mathbf{x}^* \simeq \begin{bmatrix} -0.18950 & 3.45424 & 1.32570 & -1.336679 \end{bmatrix}^\top,$
$F(\mathbf{x}^*) \simeq 7.21613 \cdot 10^{-1}$
$m=20 \quad : \ \mathbf{x}^* \simeq \begin{bmatrix} -11.5944 & 13.2036 & -0.4034 & 0.2368 \end{bmatrix}^\top,$
$F(\mathbf{x}^*) \simeq 4.29112 \cdot 10^{4}$

15. Chebyquad function

a) n variable, $m \geq n$

b) $f_i(\mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} T_i(x_j) \ - \ y_i$
where T_i is the ith Chebyshev polynomial shifted to the interval $[0,1]$ and
$y_i = \int_0^1 T_i(x)dx = \begin{cases} 0 & \text{for } i \text{ odd} \\ -1/(i^2-1) & \text{for } i \text{ even} \end{cases}$

c) $\mathbf{x}_0 = \frac{1}{n+1} \begin{bmatrix} 1 & \cdots & n \end{bmatrix}^\top, \quad \tau_0 = 1$

d)

n	$m=n$	$m=2n$
5	$F(\mathbf{x}^*) = 0$	$F(\mathbf{x}^*) \simeq 5.34479 \cdot 10^{-2}$
8	$F(\mathbf{x}^*) \simeq 1.75844 \cdot 10^{-3}$	$F(\mathbf{x}^*) \simeq 2.94780 \cdot 10^{-2}$
9	$F(\mathbf{x}^*) = 0$	$F(\mathbf{x}^*) \simeq 3.55274 \cdot 10^{-2}$
10	$F(\mathbf{x}^*) \simeq 3.25198 \cdot 10^{-3}$	$F(\mathbf{x}^*) \simeq 3.02614 \cdot 10^{-2}$

16. Brown almost linear function

a) n variable, $m=n$

b) $f_i(\mathbf{x}) = \begin{cases} x_i + x_1 + \cdots + x_n \ - \ (n+1) & i=1,\ldots n-1 \\ x_1 * \cdots * x_n \ - \ 1 & i=n \end{cases}$

c) $\mathbf{x}_0 = \frac{1}{2}\mathbf{e}, \quad \tau_0 = 1$

d) $\mathbf{x}* = \begin{bmatrix} \alpha & \cdots & \alpha & \alpha^{1-n} \end{bmatrix}^\top$, where α is a real root (in particular $\alpha=1$) in $n\alpha^n - (n+1)\alpha^{n-1} + 1 = 0$. $F(\mathbf{x}^*) = 0$.
Local minimum at $\mathbf{x}^* = \begin{bmatrix} 0 & \cdots & 0 & n+1 \end{bmatrix}^\top$. $F(\mathbf{x}^*) = 0.5$

17. Osborne 1 function

a) $n=5, \quad m=33$

b) $f_i(\mathbf{x}) = y_i - \left(x_1 + x_2 e^{-x_4 t_i} + x_3 e^{-x_5 t_i}\right)$

c) $\mathbf{x}_0 = \begin{bmatrix} 0.5 & 1.5 & -1 & 0.01 & 0.02 \end{bmatrix}^\top, \quad \tau_0 = 10^{-8}$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 0.37541 & 1.93585 & -1.46469 & 0.01287 & 0.02212 \end{bmatrix}^\top$
$F(\mathbf{x}^*) \simeq 2.73245 \cdot 10^{-5}$

Data: $t_i = 10(i-1)$,

i	y_i	i	y_i	i	y_i
1	0.000	12	110.000	23	220.000
2	10.000	13	120.000	24	230.000
3	20.000	14	130.000	25	240.000
4	30.000	15	140.000	26	250.000
5	40.000	16	150.000	27	260.000
6	50.000	17	160.000	28	270.000
7	60.000	18	170.000	29	280.000
8	70.000	19	180.000	30	290.000
9	80.000	20	190.000	31	300.000
10	90.000	21	200.000	32	310.000
11	100.000	22	210.000	33	320.000

18. *Exponential fit, 4 parameters*
   a) $n = 4, \quad m = 45$
   b) $f_i(\mathbf{x}) = y_i - \left(x_3 e^{-x_1 t_i} + x_4 e^{-x_2 t_i}\right)$
   c) $\mathbf{x}_0 = \begin{bmatrix} -1 & -2 & 1 & -1 \end{bmatrix}^\top, \quad \tau_0 = 10^{-3}$
   d) $\mathbf{x}^* \simeq \begin{bmatrix} -4 & -5 & 4 & -4 \end{bmatrix}^\top, \quad F(\mathbf{x}^*) \simeq 5{\cdot}10^{-3}$

   Data: $t_i = 0.02i$,

i	y_i	i	y_i	i	y_i
1	0.090542	16	0.288903	31	0.150874
2	0.124569	17	0.300820	32	0.126220
3	0.179367	18	0.303974	33	0.126266
4	0.195654	19	0.283987	34	0.106384
5	0.269707	20	0.262078	35	0.118923
6	0.286027	21	0.281593	36	0.091868
7	0.289892	22	0.267531	37	0.128926
8	0.317475	23	0.218926	38	0.119273
9	0.308191	24	0.225572	39	0.115997
10	0.336995	25	0.200594	40	0.105831
11	0.348371	26	0.197375	41	0.075261
12	0.321337	27	0.182440	42	0.068387
13	0.299423	28	0.183892	43	0.090823
14	0.338972	29	0.152285	44	0.085205
15	0.304763	30	0.174028	45	0.067203

19. *Exponential fit, 2 parameters*
   a) $n = 2, \quad m = 45$
   b) $f_i(\mathbf{x}) = y_i - \left(c_1(\mathbf{x}) e^{-x_1 t_i} + c_2(\mathbf{x}) e^{-x_2 t_i}\right)$
   where t_i and y_i are given in the previous problem and the coefficients $c_j(\mathbf{x})$ are found as the least squares solution to the linear problem $c_1 e^{-x_1 t_i} + c_2 e^{-x_2 t_i} \simeq y_i \;, \quad i = 1, \ldots, m$
   c) $\mathbf{x}_0 = [-1, \ -2]^\top, \quad \tau_0 = 10^{-3}$
   d) $\mathbf{x}^* \simeq [-4, \ -5]^\top, \quad F(\mathbf{x}^*) \simeq 5{\cdot}10^{-3}$

20. *Modified Meyer function*
   a) $n = 3, \quad m = 16$
   b) $f_i(\mathbf{x}) = x_1 \exp\left(\dfrac{10x_2}{t_i + x_3} - 13\right) - 10^{-3} y_i$
   where $t_i = 0.45 + 0.05i$ and the $\{y_i\}$ are given in problem 10.
   c) $\mathbf{x}_0 = \begin{bmatrix} 8.85 & 4.0 & 2.5 \end{bmatrix}^\top, \quad \tau_0 = 1$
   d) $\mathbf{x}^* \simeq \begin{bmatrix} 2.481778 & 6.18135 & 3.45224 \end{bmatrix}^\top, \quad F(\mathbf{x}^*) \simeq 43.9729{\cdot}10^{-5}$

Problems 18 – 19 were generated to simulate a data fitting problem, describing e.g. the concentration of medicine in the blood as function of time since taking the drug. The ordinates were found as $y_i = 4(e^{-4t_i} - e^{-5t_i}) + r_i$, where the $\{r_i\}$ were determined so that the given $\mathbf{x}^*$ was the true least squares solution and $F(\mathbf{x}^*) = 5{\cdot}10^{-3}$. Rounding the y_i-values to 6 decimals gives a slight perturbation in $\mathbf{x}^*$ and $F(\mathbf{x}^*)$.

References

[1] M.A. Branch and A. Grace: MATLAB *Optimization Toolbox User's Guide*. MathWorks, Natick, MA, USA. 1996.

[2] D. Hermey and G.A. Watson: *Fitting data with errors in all variables using the Huber M-estimator*. SIAM J. SCI. COMPUT. **20**, 1276 – 1298. 1999.

[3] K. Levenberg: *A method for the solution of certain problems in least squares*. Quart. Appl. Math. **2**, 164 – 168. 1944.

[4] K. Madsen: *A combined Gauss-Newton and Quasi-Newton method for non-linear least squares*. Report NI-88-10, Institute for Numerical Analysis, Technical University of Denmark (Now part of IMM, DTU). 1988.

[5] D. Marquardt: *An algorithm for least squares estimation on nonlinear parameters*. SIAM J. APPL. MATH.**11**, 431 – 441. 1963.