IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. UCTP
28.11.2000
HBN/ms

UCTP
TEST PROBLEMS FOR UNCONSTRAINED OPTIMIZATION

Hans Bruun Nielsen

TECHNICAL REPORT

IMM-REP-2000-17

IMM

UCTP – TEST PROBLEMS FOR UNCONSTRAINED OPTIMIZATION

Hans Bruun Nielsen

Contents

1. Introduction

This report documents the MATLAB and Fortran77 implementation of a series of problems for testing algorithms for unconstrained minimization, i.e. for finding

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{D}} \{F(\mathbf{x})\} \ , \tag{1.1}$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}$ and $\mathcal{D}$ is some region around $\mathbf{x}^*$, i.e. $\mathbf{x}^*$ is a local minimizer. In all the problems the function is continuously differentiable, and the subprogram also return information that can be used to compute the gradient

$$\mathbf{g}(\mathbf{x}) = \mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix} . \tag{1.2}$$

Most of the problems are least squares problems: Given a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ (with $m \geq n$). The corresponding function F is

$$F(\mathbf{x}) \equiv \tfrac{1}{2}\|\mathbf{f}(\mathbf{x})\|_2^2 = \mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) \ , \tag{1.3}$$

with the gradient

$$\mathbf{F}'(\mathbf{x}) = \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) \ , \tag{1.4a}$$

where $\mathbf{J}$ is the Jacobian matrix defined by

$$(\mathbf{J}_f(\mathbf{x}))_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}) \ . \tag{1.4b}$$

The work on this package started in connection with [5] and was further developed in connection with [6]. Work on new subroutines for the package described in [4] lead to a Fortran version. There are other testing packages around, e.g. [1], but the present package is unique for providing (as far as possible) identical test problems in MATLAB and Fortran77. This is useful for testing minimization software developed in MATLAB and translated to a production version in Fortran.

2. Matlab User's Guide

The package consists of two functions. The first is used to get the problem, and the other defines the problem for a least squares solver or a general minimizor.

2.1. User's Guide to `uctpget`

A typical call is

```
[par, x0, tau0, delta0] = uctpget(pno,m,n)
```

Input Parameters

- `pno` Problem number, cf. Section 4. Integer in the range $1 \le$ `pno` ≤ 22.
- `m, n` Number of functions and variables, respectively. Dummy variables for some of the problems.

Output Parameters

- `par` `struct` representing the problem. In all cases the field `par.p` contains the problem number, `par.pt` gives the type of the problem (see `uctpval`), and `par.xm` holds $\mathbf{x}^*$. Other fields depend on the problem.
- `x0` Starting vector.
- `tau0` Used for initial damping parameter in Levenberg-Marquardt type algorithms.
- `delta0` Used for initial radius in trust-region type algorithms.

2.2. User's Guide to `uctpval`

A typical call is

```
[F {, dF}] = uctpval(x,par)
```

Input Parameters

- `x` Argument for the test function.
- `par` `struct` representing the problem, see `uctpget`.

Output Parameters

- `F` If `par.p` ≤ 21 **and** `par.pt` $= 0$: Vector $\mathbf{f}(\mathbf{x})$.
  Otherwise: Scalar $F(\mathbf{x})$.
- `dF` Optional. If present, then
  If `par.p` ≤ 21 **and** `par.pt` $= 0$: Jacobian $\mathbf{J}(\mathbf{x})$.
  Otherwise: Gradient $\mathbf{F}'(\mathbf{x})$.

2.3. Example

The MATLAB commands

```
[par x0 tau0 delta0] = uctpget(4,2,2);  x0
```

sets parameters for the Rosenbrock function treated as a least squares problem, and returns `x0` $= [-1.2 \;\; 1]$. Next,

```
[f d] = uctpval(x0,par)
```

returns

```
f = -4.40          d =  24    10
     2.20               -1     0
```

Finally, the commands

```
par.pt = 1;  [f d] = uctpval(x0,par)
```

gives

```
f = 12.10          d = -107.80
                        -44.00
```

showing that `par.pt` $\neq 0$ changes the type of the test function.

3. Fortran User's Guide

The package consists of three subroutines. The first is used to get the problem, and the other two define the problem for a general minimizor and a least squares solver, respectively.

The calling program must include the following declarations

```
INTEGER           MM, PNO
DOUBLE PRECISION  PD(2000)
COMMON  /PROB/  PNO, MM, PD
```

PROB transfers information about the chosen problem.

The subroutines call the following BLAS (see [2]) subroutines and functions

```
DCOPY  DDOT  DNRM2  DGEMV  DSYRK
```

If BLAS is not available on your computer, you can compile uctp.f with the file minaux.f, that includes these subprograms together with a number of other BLAS routines. They were obtained from

http://www.netlib.org/blas/blas.tgz

At lines 24, 437 and 490 in the file uctp.f you can find instructions about how to modify the file if BLAS is available on your computer.

3.1. User's Guide to UCTPG

A typical call is

```
CALL UCTPG(PNO,M,N,X,TAU0,DX0,XMIN,FXMIN)
```

The parameters are

- PNO — INTEGER. Problem number, cf. Section 4. Must satisfy $1 \le$ PNO ≤ 22.
- M, N — INTEGER. Number of functions and variables, respectively. Dummy variables for some of the problems. Must be positive and must satisfy M*N ≤ 2000.
- X — REAL*8 ARRAY with N elements. Starting vector.
- TAU0 — REAL*8. Can be used for initial damping parameter in Levenberg-Marquardt type algorithms.

- DX0 — REAL*8. Can be used for initial radius in trust-region type algorithms.
- XMIN — REAL*8 ARRAY with N elements. Solution $\mathbf{x}^*$.
- FXMIN — REAL*8. Value at the solution, $F(\mathbf{x}^*)$.

3.2. User's Guide to UCTPV

A typical call is

```
CALL UCTPV(N,X,DF,F)
```

The parameters are

- N — INTEGER. Number of variables.
- X — REAL*8 ARRAY with N elements. Argument for the test function.
- DF — REAL*8 ARRAY with N elements. Returns the gradient $\mathbf{F}'$(X).
- F — REAL*8. Returns the function value F(X).

3.3. User's Guide to UCTPV2

A typical call is

```
CALL UCTPV2(N,M,X,DF,F)
```

The parameters are

- N — INTEGER. Number of variables.
- M — INTEGER. Number of functions.
- X — REAL*8 ARRAY with N elements. Argument for the test function.
- DF — REAL*8 ARRAY with M rows and N columns. Returns the Jacobian $\mathbf{J}$(X).
- F — REAL*8 ARRAY with M elements. Returns the vector $\mathbf{f}$(X).

3.4. Example

The following program (where we have omitted printing instructions) treats the Rosenbrock function.

```
      INTEGER           M,N,PNO
      DOUBLE PRECISION  X(2),XMIN(2),FXMIN,PD(2000),
     &                  TAU,DX, F1,G(2), F2(2),J(2,2)
      COMMON  /PROB/  PNO,M, PD

        CALL UCTPG(4,2,2, X,TAU,DX, XMIN,FXMIN)
        CALL UCTPV(2,X,G,F1)
        CALL UCTPV2(2,2,X,J,F2)
        STOP
      END
```

We get the following results, cf. Section 2.3. `X` $= [-1.2 \;\; 1]$

```
F1 = 12.10          G = -107.80
                         -44.00

F2 = -4.40          J =  24    10
      2.20               -1     0
```

4. Test Problems

The first 21 test problems are basically least squares problems, cf. (1.3). The first 17 problems were taken from [3], 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.

Problem 18 was 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 $F(\mathbf{x}^*)$. In problem 19 the separability [6] of the model is exploited and the number of parameters is reduced from $n=4$ to $n=2$.

Problems 20 and 21 are respectively a scaled and a "separated" version of problem 10. Finally, problem 22 is a "pure" minimization problem.

In the list below we use the following format

a) Dimension
b) Function definition
c) Starting point $\mathbf{x}_0$, τ_0 and Δ_0
d) Solution

The values τ_0 and Δ_0 are meant for use in connection with Levenberg-Marquardt and trust-region type algorithms, respectively.

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) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}, \quad \Delta_0 = 10$

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) $\mathbf{x}_0 = \mathbf{e}, \quad \tau_0 = 10^{-8}, \quad \Delta_0 = 10$

d) Any $\mathbf{x}^*$ such that $\begin{bmatrix} 1 & \cdots & n \end{bmatrix} \mathbf{x}^* = \dfrac{3}{2m+1}$.

$$F(\mathbf{x}^*) = \frac{m(m-1)}{4(2m+1)}$$

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

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

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

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

c) $\mathbf{x}_0 = \mathbf{e}$, $\tau_0 = 10^{-8}$, $\Delta_0 = 10$

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) $\mathbf{x}_0 = [-1.2,\ 1]^\top$, $\tau_0 = 1$, $\Delta_0 = 1$

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

5. *Helical valley function*

a) $n = m = 3$

b) $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 10\left(x_3 - \dfrac{5}{\pi}\arctan\dfrac{x_2}{x_1} - \theta\right) \\ 10\left(\sqrt{x_1^2 + x_2^2} - 1\right) \\ x_3 \end{bmatrix}$

with $\theta = \begin{cases} 0 & x_1 > 0 \\ 5 & x_1 < 0 \end{cases}$

c) $\mathbf{x}_0 = [-1\ \ 0\ \ 0]^\top$, $\tau_0 = 1$, $\Delta_0 = 1$

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

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) $\mathbf{x}_0 = [3\ \ -1\ \ 0\ \ 1]^\top$, $\tau_0 = 10^{-8}$, $\Delta_0 = 1$

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

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) $\mathbf{x}_0 = [0.5,\ -2]^\top$, $\tau_0 = 1$, $\Delta_0 = 1$

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

8. *Bard function*

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

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

c) $\mathbf{x}_0 = \mathbf{e}$, $\tau_0 = 10^{-8}$, $\Delta_0 = 1$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 0.082411 & 1.133036 & 2.343695 \end{bmatrix}^\top$,

$F(\mathbf{x}^*) \simeq 4.10744 \cdot 10^{-3}$

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) $\mathbf{x}_0 = \left[\,0.25 \quad 0.39 \quad 0.415 \quad 0.39\,\right]^\top, \quad \tau_0 = 1, \quad \Delta_0 = 0.1$

d) $\mathbf{x}^* \simeq \left[\,0.192807 \quad 0.191282 \quad 0.123057 \quad 0.136062\,\right]^\top,$
$F(\mathbf{x}^*) \simeq 1.53753 \cdot 10^{-4}$

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) $\mathbf{x}_0 = \left[\,0.02 \quad 4000 \quad 250\,\right]^\top, \quad \tau_0 = 1, \quad \Delta_0 = 100$

d) $\mathbf{x}^* \simeq \left[\,0.00560964 \quad 6181.35 \quad 345.224\,\right]^\top, \quad F(\mathbf{x}^*) \simeq 43.9729$

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) $\mathbf{x}_0 = \mathbf{0}\,, \quad \tau_0 = 10^{-8}, \quad \Delta_0 = 1$

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) $n = 3, \quad m \ge 3$ 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) $\mathbf{x}_0 = \left[\,0 \quad 10 \quad 20\,\right]^\top, \quad \tau_0 = 10^{-8}, \quad \Delta_0 = 1, \quad \Delta_0 = 1$

d) $\mathbf{x}^* = \left[\,1 \quad 10 \quad 1\,\right]^\top, \;\; \mathbf{x}^* = \left[\,10 \quad 1 \quad -1\,\right]^\top$ or
$\mathbf{x}^* = \left[\,\alpha \quad \alpha \quad 0\,\right]^\top$ for any $\alpha \in \mathbb{R}$. $F(\mathbf{x}^*) = 0$

13. Jennrich and Sampson function

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

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

c) $\mathbf{x}_0 = [0.3,\ 0.4]^\top, \quad \tau_0 = 1, \quad \Delta_0 = 0.05$

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

14. Brown and Dennis function

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

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

c) $\mathbf{x}_0 = \begin{bmatrix} 25 & 5 & -5 & -1 \end{bmatrix}^\top, \quad \tau_0 = 10^{-3}, \quad \Delta_0 = 0.5$

d) $m=5$: $\mathbf{x}^* \simeq \begin{bmatrix} 0.77781 & 1.87187 & 1.15301 & -0.67095 \end{bmatrix}^\top$, $F(\mathbf{x}^*) \simeq 9.08309 \cdot 10^{-5}$

$m=10$: $\mathbf{x}^* \simeq \begin{bmatrix} -0.18950 & 3.45424 & 1.32570 & -1.336679 \end{bmatrix}^\top$, $F(\mathbf{x}^*) \simeq 7.21613 \cdot 10^{-1}$

$m=20$: $\mathbf{x}^* \simeq \begin{bmatrix} -11.5944 & 13.2036 & -0.4034 & 0.2368 \end{bmatrix}^\top$, $F(\mathbf{x}^*) \simeq 4.29112 \cdot 10^{4}$

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) $\mathbf{x}_0 = \frac{1}{n+1} \begin{bmatrix} 1 & \cdots & n \end{bmatrix}^\top, \quad \tau_0 = 1, \quad \Delta_0 = \frac{1}{n+1}$

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) $\mathbf{x}_0 = \frac{1}{2}\mathbf{e}, \quad \tau_0 = 1, \quad \Delta_0 = 1$

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

c) $\mathbf{x}_0 = \begin{bmatrix} 0.5 & 1.5 & -1 & 0.01 & 0.02 \end{bmatrix}^\top, \quad \tau_0 = 10^{-8}, \quad \Delta_0 = 0.1$

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	0.844	2	0.908	3	0.932
4	0.936	5	0.925	6	0.908
7	0.881	8	0.850	9	0.818
10	0.784	11	0.751	12	0.718
13	0.685	14	0.658	15	0.628
16	0.603	17	0.580	18	0.558
19	0.538	20	0.522	21	0.506
22	0.490	23	0.478	24	0.467
25	0.457	26	0.448	27	0.438

18. Exponential fit, 4 parameters

a) $n=4, \quad m=45$

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

c) $\mathbf{x}_0 = \begin{bmatrix} -1 & -2 & 1 & -1 \end{bmatrix}^\top, \quad \tau_0 = 10^{-3}, \quad \Delta_0 = 1$

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) $f_i(\mathbf{x}) = y_i - (c_1(\mathbf{x}) e^{x_1 t_i} + c_2(\mathbf{x}) e^{x_2 t_i})$

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) $\mathbf{x}_0 = [-1,\ -2]^\top, \quad \tau_0 = 10^{-3}, \quad \Delta_0 = 1$

d) $\mathbf{x}^* \simeq [-4,\ -5]^\top, \quad F(\mathbf{x}^*) \simeq 5{\cdot}10^{-3}$

20. Scaled 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) $\mathbf{x}_0 = \begin{bmatrix} 8.85 & 4.0 & 2.5 \end{bmatrix}^\top, \quad \tau_0 = 1, \quad \Delta_0 = 1$

d) $\mathbf{x}^* \simeq \begin{bmatrix} 2.481778 & 6.18135 & 3.45224 \end{bmatrix}^\top,$
$F(\mathbf{x}^*) \simeq 43.9729{\cdot}10^{-6}$

21. Separated Meyer function

a) $n=2, \quad m=16$

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

where $t_i = 45 + 5i$ and the $\{y_i\}$ are given in problem 10. The coefficient $c(\mathbf{x})$ is found as the least squares solution to the linear problem $c \ \exp\left(\frac{x_1}{t_i + x_2}\right) \simeq y_i, \ i=1,\ldots,m$

c) $\mathbf{x}_0 = \begin{bmatrix} 4000 & 250 \end{bmatrix}^\top, \quad \tau_0 = 1, \quad \Delta_0 = 100$

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

22. Exp and squares

a) n variable, $\quad m=1$

b) $F(\mathbf{x}) = \exp\left(-\sum_{j=1}^{n} x_j\right) + \frac{1}{2} \sum_{j=1}^{n} j^2 x_j^2$

c) $\mathbf{x}_0 = \mathbf{0}, \quad \tau_0 = 10^{-3}, \quad \Delta_0 = 1$

d) $x_j^* = e^{-y}/j^2$ where $y = y_n$ is the solution to $(1 + \cdots \frac{1}{n^2}) e^{-y} = y$

References

[1] I. Bongartz, A.R. Conn, N. Gould and Ph.L. Toint: *CUTE: constrained and unconstrained testing environment,* ACM Trans. Math. Soft. **21**, 1995, 123–160. An enhanced test set is available at
http://www.rl.ac.uk/departments/ccd/numerical/cute/mastsif.html

[2] J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart: *An Extended Set of Fortran Basic Linear Algebra Subprograms,* ACM Trans. Math. Soft. **14**, 1988, 1–17.

[3] 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.

[4] K. Madsen, O. Tingleff, P.C. Hansen and W. Owczarz (1990): *Robust Subroutines for Non-Linear Optimization.* Report NI-90-06, Institute for Numerical Analysis (now part of IMM), Technical University of Denmark.

[5] H.B. Nielsen: *Damping Parameter in Marquardt's Method.* Report IMM-REP-1999-05, IMM, DTU. Available at
http://www.imm.dtu.dk/~hbn/publ/TR9905.ps.Z

[6] H.B. Nielsen: *Separable Nonlinear Least Squares.* IMM, DTU. Report IMM-REP-2000-01. Available at
http://www.imm.dtu.dk/~hbn/publ/TR0001.ps.Z