Use of Least Squares Software

The Matlab-functions **Marquardt**, **SMarquardt** and **KMhybrid** can be used to find

$$\mathbf{x}^* = \mathrm{argmin}\{F(\mathbf{x}) \equiv \tfrac{1}{2}\mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})\} \ ,$$

where $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a given function. The Matlab-functions **DogLeg** and **SDogLeg** solve the problem in the special case where $m = n$ and $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, i.e. $\mathbf{x}^*$ solves a nonlinear system of equations. The functions **SMarquardt** and **SDogLeg** use function values only, while the other functions also demand the Jacobian $\mathbf{J}_f(\mathbf{x})$.

As examples of the use of the Matlab-functions we shall consider the *Rosenbrock function* $\mathbf{f} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ defined by

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

and the *Modified Rosenbrock function* $\mathrm{g} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ defined by

$$\mathrm{g}(\mathbf{x}) = \begin{bmatrix} 10 * (x_2 - x_1^2) \\ 1 - x_1 \\ \lambda \end{bmatrix} \ ,$$

where λ is some constant. Both functions have the minimizer $\mathbf{x}^* = [1,\ 1]^\top$ with $\mathbf{f}(\mathbf{x}*) = [0,\ 0]^\top$, $\mathrm{g}(\mathbf{x}^*) = [0,\ 0,\ \lambda]^\top$.

We assume that these two functions are defined in the files ros.m and mros.m with contents respectively

```
function [f, J] = ros(x, p)
  f = [10*(x(2) - x(1)^2); 1-x(1)];
  if nargout > 1, J = [-20*x(1) 10; -1 0]; end
```

and

```
function [f, J] = mros(x, p)
  f = [10*(x(2) - x(1)^2); 1-x(1); p];
  if nargout > 1, J = [-20*x(1) 10; -1 0; 0 0]; end
```


The following Matlab programme finds $\mathbf{x}^*$ from the starting point $\mathbf{x}_0 = [-1.2,\ 1]^\top$.

```
xe = [1; 1];  x0 = [-1.2 1];
disp('Marquardt:'),  tau = [1 1e-10 1e-12 100];
[X info] = Marquardt('ros',[],x0,tau)
erx = norm(X - xe)
disp('SMarquardt:')
[X info] = SMarquardt('ros',[],x0,[tau 1e-8]);
erx = norm(X - xe)
its = info(5),  st = info(6),  neval = info(7)
```

The elements in **tau** are used to find the initial Marquardt damping parameter

$$\mu_0 = \mathtt{tau}(1) \cdot \max\{(\mathbf{J}_f(\mathbf{x}_0)^\top \mathbf{J}_f(\mathbf{x}_0))_{ii}\}$$

and in the stopping criteria

$$\|\mathbf{F}'(\mathbf{x}_k)\|_\infty \le \mathtt{tau}(2) \ , \tag{1a}$$

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 \le \mathtt{tau}(3)(\mathtt{tau}(3) + \|\mathbf{x}_l\|_2) \ , \tag{1b}$$

$$k \ge \mathtt{tau}(4) \ . \tag{1c}$$

In **SMarquardt** the parameter **tau** must have 5 elements, where $\delta = \mathtt{tau}(5)$ is used as steplength in the difference approximation to $\mathbf{J}_f(\mathbf{x})$.

The above programme gives the following results,

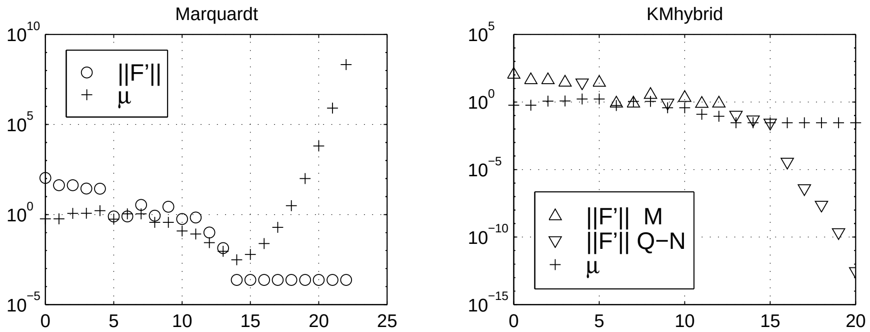
```
Marquardt:    erx = 1.7271e-10
info = 2.9782e-21 3.0942e-11 6.9975e-08 4.0969e-07 2.4000e+01 1.0000e+00

SMarquardt:  erx = 1.4056e-13   its = 39   st = 1   neval = 48
```

This shows that **Marquardt** is stopped by (1a) after 24 iteration steps, i.e. 25 evaluations of $\mathbf{f}$ and $\mathbf{J}_f$. At that point $\mathbf{F}' \simeq 3.09{\cdot}10^{-11} < \mathtt{tau}(2)$. **SMarquardt** is stopped by (1a) after 39 iteration steps involving a total of 48 evaluations of $\mathbf{f}$.

The next programme illustratess use of the trace option and the function **KMhybrid**.

```
x0 = [-1.2 1];  lam = 1e4;  tau = [1e-3 1e-10 1e-12 100];
[X info p] = Marquardt('ros1',lam,x0,tau);
erxM = norm(X(:,end) - [1;1]),  st = info(6)
clf, subplot(2,2,1),  t = 0 :  size(p,2)-1;
semilogy(t,p(2,:),'o', t,p(3,:),'+'),  grid on
title('Marquardt'), legend('||g||', '\mu',2)
[X info p] = KMhybrid('ros1',lam,x0,tau);
erxH = norm(X(:,end) - [1;1]),  st = info(6)
subplot(2,2,2),  t = 0 :  size(p,2)-1;
m1 = find(p(4,:)  == 1); m2 = find(p(4,:)  == 2);
semilogy(t(m1),p(2,m1),'^', t(m2),p(2,m2),'v', t,p(3,:),'+')
grid on
title('KMhybrid'), legend('||g|| M ', '||g|| Q-N','\mu',3)
```

Marquardt is stopped by (1b) with **erxM** = **1.1841e-04**, while **KMhybrid** is stopped by (1a) with **erxH** = **5.7648e-14**. The plot is given below. In the right-hand figure "M" and "Q-N" is used to denote that the iterate was computed by a Marquardt step and a Quasi-Newton step, respectively.



Finally, we illustrate the use of **DogLeg** and **SDogLeg** by the following programme,

```
xe = [1; 1];  x0 = [-1.2 1];
disp('DogLeg:'),  tau = [.1 1e-10 1e-12 1e-16 100];
[X info] = DogLeg('ros',[],x0,tau);
erx = norm(X - xe),
disp('SDogLeg:')
[X info] = SDogLeg('ros',[],x0,[tau 1e-8]);
erx = norm(X - xe)
its = info(5),  st = info(6),  neval = info(7)
```

The first element in **tau** is the initial trust region radius, while the other elements are used in the stopping criteria

$$\|\mathbf{F}'(\mathbf{x}_k)\|_\infty \le \mathtt{tau}(2) \ , \tag{2a}$$

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 \le \mathtt{tau}(3)(\mathtt{tau}(3) + \|\mathbf{x}_l\|_2) \ , \tag{2b}$$

$$\|\mathbf{f}(\mathbf{x}_k)\|_\infty \le \mathtt{tau}(4) \ , \tag{2c}$$

$$k \ge \mathtt{tau}(5) \ . \tag{2d}$$

Except for (2c) these criteria are identical with (1). In **SDogLeg** the parameter **tau** must have 6 elements, where $\delta = \mathtt{tau}(6)$ is used as steplength in the difference approximation to $\mathbf{J}_f(\mathbf{x})$.

The above programme gives the following results,

```
DogLeg:    erx = 0
info = 0 0 6.2052e-03 1.2895e+00 1.6000e+01 1.0000e+00

SDogLeg:    erx = 0  its = 29   st = 1   neval = 35
```

This shows that both functions are stopped by (2a) at the true solution. **DogLeg** needs 17 evaluations of $\mathbf{f}$ and $\mathbf{J}_f$, and **SDogLeg** uses 29 iteration steps involving a total of 35 evaluations of $\mathbf{f}$.