IMM

DTU

METHODS FOR NON-LINEAR LEAST SQUARES PROBLEMS

2nd Edition, April 2004

K. Madsen, H.B. Nielsen, O. Tingleff

Informatics and Mathematical Modelling
Technical University of Denmark

CONTENTS

1. Introduction and Definitions

In this booklet we consider the following problem,

Definition 1.1. Least Squares Problem

Find $\mathbf{x}^*$, a local minimizer for[1)]

$$F(\mathbf{x}) \;=\; \tfrac{1}{2}\sum_{i=1}^{m}\left(f_i(\mathbf{x})\right)^2 \;,$$

where $f_i:\ \mathbb{R}^n \mapsto \mathbb{R},\ i=1,\ldots,m$ are given functions, and $m \geq n$.

Example 1.1. An important source of least squares problems is *data fitting*. As an example consider the *data points* $(t_1,y_1),\ldots,(t_m,y_m)$ shown below

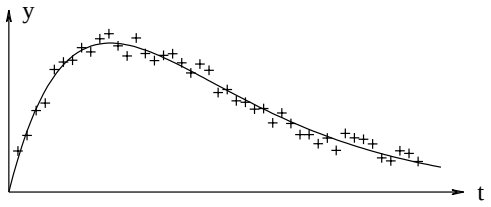

Figure 1.1. *Data points* $\{(t_i,y_i)\}$ *(marked by +) and model* $M(\mathbf{x},t)$ *(marked by full line.)*

Further, we are given a *fitting model*,

$$M(\mathbf{x},t) = x_3 e^{x_1 t} + x_4 e^{x_2 t} \;.$$

[1)] The factor $\frac{1}{2}$ in the definition of $F(\mathbf{x})$ has no effect on $\mathbf{x}^*$. It is introduced for convenience, see page 18.

The model depends on the *parameters* $\mathbf{x} = [x_1,x_2,x_3,x_4]^\top$. We assume that there exists an $\mathbf{x}^\dagger$ so that

$$y_i = M(\mathbf{x}^\dagger,t_i) + \varepsilon_i \;,$$

where the $\{\varepsilon_i\}$ are (measurement) errors on the data ordinates, assumed to behave like "white noise".

For any choice of $\mathbf{x}$ we can compute the *residuals*

$$\begin{aligned} f_i(\mathbf{x}) \;&=\; y_i - M(\mathbf{x},t_i) \\ &=\; y_i - x_3 e^{x_1 t_i} - x_4 e^{x_2 t_i} \;, \quad i=1,\ldots,m \;. \end{aligned}$$

For a *least squares fit* the parameters are determined as the minimizer $\mathbf{x}^*$ of the sum of squared residuals. This is seen to be a problem of the form in Definition 1.1 with $n=4$. The graph of $M(\mathbf{x}^*,t)$ is shown by full line in Figure 1.1. ∎

A least squares problem is a special variant of the more general problem: Given a function F: $\mathbb{R}^n \mapsto \mathbb{R}$, find an argument of F that gives the minimum value of this so-called *objective function* or *cost function*.

Definition 1.2. Global Minimizer

Given $F:\ \mathbb{R}^n \mapsto \mathbb{R}$. Find

$$\mathbf{x}^+ \;=\; \operatorname{argmin}_{\mathbf{x}}\{F(\mathbf{x})\} \;.$$

This problem is very hard to solve in general, and we only present methods for solving the simpler problem of finding a local minimizer for F, an argument vector which gives a minimum value of F inside a certain region whose size is given by δ, where δ is a small, positive number.

Definition 1.3. Local Minimizer

Given $F:\ \mathbb{R}^n \mapsto \mathbb{R}$. Find $\mathbf{x}^*$ so that

$$F(\mathbf{x}^*) \leq F(\mathbf{x}) \quad \text{for} \quad \|\mathbf{x}-\mathbf{x}^*\| < \delta \;.$$

In the remainder of this introduction we shall discuss some basic concepts in optimization, and Chapter 2 is a brief review of methods for finding a local

minimizer for general cost functions. For more details we refer to Frandsen et al (2004). In Chapter 3 we give methods that are specially tuned for least squares problems.

We assume that the cost function F is differentiable and so smooth that the following *Taylor expansion* is valid,[2)]

$$F(\mathbf{x}+\mathbf{h}) \;=\; F(\mathbf{x}) + \mathbf{h}^\top \mathbf{g} + \tfrac{1}{2}\mathbf{h}^\top \mathbf{H}\,\mathbf{h} + O(\|\mathbf{h}\|^3)\;, \tag{1.4a}$$

where $\mathbf{g}$ is the *gradient*,

h: downhill direction, a some increment vector of x.

F is a scalar-valued function; its gradient g is a vector-valued function.

$$\mathbf{g} \;\equiv\; \mathbf{F}'(\mathbf{x}) \;=\; \begin{bmatrix} \dfrac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \dfrac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix}\;, \tag{1.4b}$$

and $\mathbf{H}$ is the *Hessian*,

$$\mathbf{H} \;\equiv\; \mathbf{F}''(\mathbf{x}) \;=\; \left[\frac{\partial^2 F}{\partial x_i \partial x_j}(\mathbf{x})\right]\;. \tag{1.4c}$$

If $\mathbf{x}^*$ is a local minimizer and $\|\mathbf{h}\|$ is sufficiently small, then we cannot find a point $\mathbf{x}^*+\mathbf{h}$ with a smaller F-value. Combining this observation with (1.4a) we get

Theorem 1.5. Necessary condition for a local minimizer.
If $\mathbf{x}^*$ is a local minimizer, then

$$\mathbf{g}^* \;\equiv\; \mathbf{F}'(\mathbf{x}^*) \;=\; \mathbf{0}\;.$$

We use a special name for arguments that satisfy the necessary condition:

Definition 1.6. Stationary point. If

$$\mathbf{g}_s \;\equiv\; \mathbf{F}'(\mathbf{x}_s) \;=\; \mathbf{0}\;,$$

then $\mathbf{x}_s$ is said to be a *stationary point* for F.

[2)] Unless otherwise specified, $\|\cdot\|$ denotes the 2-norm, $\|\mathbf{h}\| = \sqrt{h_1^2 + \cdots + h_n^2}$.

the Euclidean distance

Thus, a local minimizer is also a stationary point, but so is a local maximizer. A stationary point which is neither a local maximizer nor a local minimizer is called a *saddle point*. In order to determine whether a given stationary point is a local minimizer or not, we need to include the second order term in the Taylor series (1.4a). Inserting $\mathbf{x}_s$ we see that

$$F(\mathbf{x}_s+\mathbf{h}) = F(\mathbf{x}_s) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{H}_s\,\mathbf{h} + O(\|\mathbf{h}\|^3) \qquad \text{with } \mathbf{H}_s = \mathbf{F}''(\mathbf{x}_s)\;. \tag{1.7}$$

From definition (1.4c) of the Hessian it follows that any $\mathbf{H}$ is a symmetric matrix. If we request that $\mathbf{H}_s$ is *positive definite*, then its eigenvalues are greater than some number $\delta > 0$ (see Appendix A), and

not exactly true.

$$\mathbf{h}^\top \mathbf{H}_s\,\mathbf{h} > \delta\,\|\mathbf{h}\|^2\;.$$

This shows that for $\|\mathbf{h}\|$ sufficiently small the third term on the right-hand side of (1.7) will be dominated by the second. This term is positive, so that we get

Theorem 1.8. Sufficient condition for a local minimizer.
Assume that $\mathbf{x}_s$ is a stationary point and that $\mathbf{F}''(\mathbf{x}_s)$ is positive definite. Then $\mathbf{x}_s$ is a local minimizer.

If $\mathbf{H}_s$ is *negative definite*, then $\mathbf{x}_s$ is a local maximizer. If $\mathbf{H}_s$ is *indefinite* (ie it has both positive and negative eigenvalues), then $\mathbf{x}_s$ is a saddle point.

2. Descent Methods

All methods for non-linear optimization are iterative: From a starting point $\mathbf{x}_0$ the method produces a series of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$, which (hopefully) converges to $\mathbf{x}^*$, a local minimizer for the given function, see Definition 1.3. Most methods have measures which enforce the *descending condition*

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k) \ . \tag{2.1}$$

This prevents convergence to a maximizer and also makes it less probable that we converge towards a saddle point. If the given function has several minimizers the result will depend on the starting point $\mathbf{x}_0$. We do not know which of the minimizers that will be found; it is not necessarily the minimizer closest to $\mathbf{x}_0$.

In many cases the method produces vectors which converge towards the minimizer in two clearly different stages. When $\mathbf{x}_0$ is far from the solution we want the method to produce iterates which move steadily towards $\mathbf{x}^*$. In this "global stage" of the iteration we are satisfied if the errors do not increase except in the very first steps, ie

$$\|\mathbf{e}_{k+1}\| < \|\mathbf{e}_k\| \quad \text{for } k > K,$$ K denotes the very first steps ?

where $\mathbf{e}_k$ denotes the current error,

$$\mathbf{e}_k \ = \ \mathbf{x}_k - \mathbf{x}^* \ . \tag{2.2}$$

In the final stage of the iteration, where $\mathbf{x}_k$ is close to $\mathbf{x}^*$, we want faster convergence. We distinguish between

Linear convergence:

$$\|\mathbf{e}_{k+1}\| \le a\|\mathbf{e}_k\| \qquad \text{when } \|\mathbf{e}_k\| \text{ is small; } \ 0 < a < 1 \ , \tag{2.3a}$$

Quadratic convergence:

$$\|\mathbf{e}_{k+1}\| = O(\|\mathbf{e}_k\|^2) \qquad \text{when } \|\mathbf{e}_k\| \text{ is small} \ , \tag{2.3b}$$

Superlinear convergence:

$$\|\mathbf{e}_{k+1}\|/\|\mathbf{e}_k\| \to 0 \qquad \text{for } k \to \infty \ . \tag{2.3c}$$

The methods presented in this lecture note are descent methods which satisfy the descending condition (2.1) in each step of the iteration. One step from the current iterate consists in

1. Find a descent direction $\mathbf{h}_d$ (discussed below), and
2. find a step length giving a good decrease in the F-value.

Thus an outline of a descent method is

Algorithm 2.4. Descent method

begin
  $k := 0$; $\mathbf{x} := \mathbf{x}_0$; *found* := **false** {Starting point}
  while (**not** *found*) **and** $(k < k_{\max})$
    $\mathbf{h}_d :=$ search_direction($\mathbf{x}$) {From $\mathbf{x}$ and downhill}
    if (no such $\mathbf{h}$ exists)
      found := **true** {$\mathbf{x}$ is stationary}
    else
      $\alpha :=$ step_length($\mathbf{x}, \mathbf{h}_d$) {from $\mathbf{x}$ in direction $\mathbf{h}_d$}
      $\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_d$; $k := k+1$ {next iterate}
end

Consider the variation of the F-value along the half line starting at $\mathbf{x}$ and with direction $\mathbf{h}$. From the Taylor expansion (1.4a) we see that

$$\begin{aligned} F(\mathbf{x}+\alpha\mathbf{h}) &= F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) + O(\alpha^2) \\ &\simeq F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) \quad \text{for } \alpha \text{ sufficiently small.} \end{aligned} \tag{2.5}$$

We say that $\mathbf{h}$ is a *descent direction* if $F(\mathbf{x}+\alpha\mathbf{h})$ is a decreasing function of α at $\alpha = 0$. This leads to the following definition.

Definition 2.6. Descent direction.

$\mathbf{h}$ is a descent direction for F at $\mathbf{x}$ if $\mathbf{h}^\top \mathbf{F}'(\mathbf{x}) < 0$.

If no such $\mathbf{h}$ exists, then $\mathbf{F}'(\mathbf{x}) = \mathbf{0}$, showing that in this case $\mathbf{x}$ is stationary. Otherwise, we have to choose α, ie how far we should go from $\mathbf{x}$ in the direction given by $\mathbf{h}_{\rm d}$, so that we get a decrease in the value of the objective function. One way of doing this is to find (an approximation to)

$$\alpha_{\rm e} \;=\; \text{argmin}_{\alpha>0}\{F(\mathbf{x}+\alpha\mathbf{h})\}\ . \qquad (2.7)$$

with x and h fixed

The process is called *line search*, and is discussed in Section 2.3. First, however, we shall introduce two methods for computing a descent direction.

2.1. The Steepest Descent method

positive scalar a; "negative" vector h.

From (2.5) we see that when we perform a step $\alpha\mathbf{h}$ with positive α, then the relative gain in function value satisfies

projecting F'(x) onto the unit vector denoted by h^T / norm(h)

$$\lim_{\alpha\to 0} \frac{F(\mathbf{x}) - F(\mathbf{x}+\alpha\mathbf{h})}{\alpha\|\mathbf{h}\|} = -\frac{1}{\|\mathbf{h}\|}\,\mathbf{h}^\top \mathbf{F}'(\mathbf{x}) = -\|\mathbf{F}'(\mathbf{x})\|\cos\theta\ ,$$

where θ is the angle between the vectors $\mathbf{h}$ and $\mathbf{F}'(\mathbf{x})$. This shows that we get the greatest gain rate if $\theta = \pi$, ie if we use the steepest descent direction $\mathbf{h}_{\rm sd}$ given by

$$\mathbf{h}_{\rm sd} = -\mathbf{F}'(\mathbf{x})\ . \qquad (2.8)$$

The method based on (2.8) (ie $\mathbf{h}_{\rm d} = \mathbf{h}_{\rm sd}$ in Algorithm 2.4) is called the *steepest descent method* or *gradient method*. The choice of descent direction is "the best" (locally) and we could combine it with an exact line search (2.7). A method like this converges, but the final convergence is linear and often very slow. Examples in Frandsen et al (2004) show how the steepest descent method with exact line search and finite computer precision can fail to find the minimizer of a second degree polynomial. For many problems, however, the method has quite good performance in the initial stage of the iterative process.

Considerations like this has lead to the so-called *hybrid methods*, which – as the name suggests – are based on two different methods. One which is good in the initial stage, like the gradient method, and another method which is good in the final stage, like Newton's method; see the next section. A major problem with a hybrid method is the mechanism which switches between the two methods when appropriate.

as the Levenberg-Marquardt method discussed in Section 3.2

2.2. Newton's Method

We can derive this method from the condition that $\mathbf{x}^*$ is a stationary point. According to Definition 1.6 it satisfies $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$. This is a nonlinear system of equations, and from the Taylor expansion

$$\begin{aligned}\mathbf{F}'(\mathbf{x}+\mathbf{h}) \;&=\; \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2)\\ &\simeq\; \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} \quad \text{for } \|\mathbf{h}\| \text{ sufficiently small}\end{aligned}$$

we derive *Newton's method:* Find $\mathbf{h}_{\rm n}$ as the solutions to

$$\mathbf{H}\,\mathbf{h}_{\rm n} = -\mathbf{F}'(\mathbf{x}) \quad \text{with } \mathbf{H} = \mathbf{F}''(\mathbf{x})\ , \qquad (2.9a)$$

and compute the next iterate by

$$\mathbf{x} := \mathbf{x} + \mathbf{h}_{\rm n}\ . \qquad (2.9b)$$

Suppose that $\mathbf{H}$ is positive definite, then it is nonsingular (implying that (2.9a) has a unique solution), and $\mathbf{u}^\top\mathbf{H}\,\mathbf{u} > 0$ for all nonzero $\mathbf{u}$. Thus, by multiplying with $\mathbf{h}_{\rm n}^\top$ on both sides of (2.9a) we get

$$0 < \mathbf{h}_{\rm n}^\top\mathbf{H}\,\mathbf{h}_{\rm n} = -\mathbf{h}_{\rm n}^\top\mathbf{F}'(\mathbf{x})\ , \qquad (2.10)$$

showing that $\mathbf{h}_{\rm n}$ is a descent direction: it satisfies the condition in Definition 2.6.

Newton's method is very good in the final stage of the iteration, where $\mathbf{x}$ is close to $\mathbf{x}^*$. One can show (see Frandsen et al (2004)) that if the Hessian at the solution is positive definite (the sufficient condition in Theorem 1.8 is satisfied) and if we are at a position inside the region around $\mathbf{x}^*$ where

$\mathbf{F}''(\mathbf{x})$ is positive definite, then we get quadratic convergence (defined in (2.3)). On the other hand, if $\mathbf{x}$ is in a region where $\mathbf{F}''(\mathbf{x})$ is negative definite everywhere, and where there is a stationary point, the basic Newton method (2.9) would converge (quadratically) towards this stationary point, which is a maximizer. We can avoid this by requiring that all steps taken are in descent directions.

We can build a hybrid method, based on Newton's method and the steepest descent method. According to (2.10) the Newton step is guaranteed to be downhill if $\mathbf{F}''(\mathbf{x})$ is positive definite, so a sketch of the central section of this hybrid algorithm could be

$$\begin{aligned}
&\textbf{if } \mathbf{F}''(\mathbf{x}) \text{ is positive definite} \\
&\quad \mathbf{h} := \mathbf{h}_{\mathrm{n}} \\
&\textbf{else} \\
&\quad \mathbf{h} := \mathbf{h}_{\mathrm{sd}} \\
&\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}
\end{aligned} \tag{2.11}$$

Here, $\mathbf{h}_{\mathrm{sd}}$ is the steepest descent direction and α is found by line search; see Section 2.3. A good tool for checking a matrix for positive definiteness is Cholesky's method (see Appendix A) which, when successful, is also used for solving the linear system in question. Thus, the check for definiteness is almost for free.

In Section 2.4 we introduce some methods, where the computation of the search direction $\mathbf{h}_{\mathrm{d}}$ and step length α is done simultaneously, and give a version of (2.11) without line search. Such hybrid methods can be very efficient, but they are hardly ever used. The reason is that they need an implementation of $\mathbf{F}''(\mathbf{x})$, and for complicated application problems this is not available. Instead we can use a so-called *Quasi-Newton method*, based on series of matrices which gradually approach $\mathbf{H}^* = \mathbf{F}''(\mathbf{x}^*)$. In Section 3.4 we present such a method. Also see Chapter 5 in Frandsen et al (2004).

2.3. Line Search

Given a point $\mathbf{x}$ and a descent direction $\mathbf{h}$. The next iteration step is a move from $\mathbf{x}$ in direction $\mathbf{h}$. To find out, how far to move, we study the variation of the given function along the half line from $\mathbf{x}$ in the direction $\mathbf{h}$,

$$\varphi(\alpha) = F(\mathbf{x}+\alpha\mathbf{h}) \,, \quad \mathbf{x} \text{ and } \mathbf{h} \text{ fixed}, \ \alpha \geq 0 \,. \tag{2.12}$$

An example of the behaviour of $\varphi(\alpha)$ is shown in Figure 2.1.

\alpha = 0, equivalent to the F(x);
\alpha != 0, equivalent to the F(x+\alpha*h),
which is the next iteration.
That's how it shows the variation of the cost function

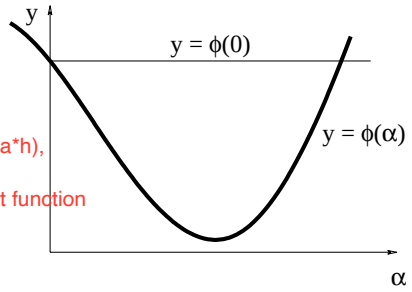

Figure 2.1. *Variation of the cost function along the search line.*

Our $\mathbf{h}$ being a descent direction ensures that

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

indicating that if α is sufficiently small, we satisfy the descending condition (2.1), which is equivalent to

$$\varphi(\alpha) < \varphi(0) \,.$$

Often, we are given an initial guess on α, eg $\alpha = 1$ with Newton's method. Figure 2.1 illustrates that three different situations can arise

1° α is so small that the gain in value of the objective function is very small. α should be increased.

2° α is too large: $\varphi(\alpha) \geq \varphi(0)$. Decrease α in order to satisfy the descent condition (2.1).

3° α is close to the minimizer[1)] of $\varphi(\alpha)$. Accept this α-value.

[1)] More precisely: the smallest local minimizer of φ. If we increase α beyond the interval shown in Figure 2.1, it may well happen that we get close to another local minimum for F.

An *exact line search* is an iterative process producing a series $\alpha_1, \alpha_2 \ldots$. The aim is to find the true minimizer α_e defined in (2.7), and the algorithm stops when the iterate α_s satisfies

approximately arrived the stationary point at which the derivative is close to zero and way less to the starting point of this line search.

$$|\varphi'(\alpha_s)| \le \tau\,|\varphi'(0)|\,,$$

where τ is a small, positive number. In the iteration we can use approximations to the variation of $\varphi(\alpha)$ based on the computed values of

$$\varphi(\alpha_k) = F(\mathbf{x}+\alpha_k\mathbf{h}) \quad \text{and} \quad \varphi'(\alpha_k) = \mathbf{h}^\top \mathbf{F}'(\mathbf{x}+\alpha_k\mathbf{h})\,.$$

See Sections 2.5 – 2.6 in Frandsen et al (2004) for details.

Exact line search can waste much computing time: When $\mathbf{x}$ is far from $\mathbf{x}^*$ the search direction $\mathbf{h}$ may be far from the direction $\mathbf{x}^*-\mathbf{x}$, and there is no need to find the true minimum of φ very accurately. This is the background for the so-called *soft line search*, where we accept an α-value if it does not fall in the categories 1° or 2° listed above. We use a stricter version of the descending condition (2.1), viz

这个condition的相反：确保找到的点不会在离起点很近的的位置。当 x* 离 x较远时，这样可以加快 search speed.

$$\varphi(\alpha_s) \le \varphi(0) + \gamma_1\cdot\varphi'(0)\cdot\alpha \quad \text{with } 0<\gamma_1<1\,. \tag{2.13a}$$

This ensures that we are not in case 2°. Case 1° corresponds to the point $(\alpha, \varphi(\alpha))$ being too close to the starting tangent, and we supplement with the condition

这个condition的相反：导数比起点小，确保比起点更接近 local mimina

$$\varphi'(\alpha_s) \ge \gamma_2\cdot\varphi'(0) \quad \text{with } \gamma_1<\gamma_2<1\,. \tag{2.13b}$$

If the starting guess on α satisfies both these criteria, then we accept it as α_s. Otherwise, we have to iterate as sketched for exact line search. Details can be seen in Section 2.5 of Frandsen et al (2004).

radius of the trust region 和 damping factor 可以看作是两个作用相反的量。

2.4. Trust Region and Damped Methods

Assume that we have a *model* L of the behaviour of F in the neighbourhood of the current iterate $\mathbf{x}$,

The neighbourhood means: consider the h with the fixed starting point x. It makes us concentrate on the behaviour of the iterate step h.

$$F(\mathbf{x}+\mathbf{h}) \;\simeq\; L(\mathbf{h}) \;\equiv\; F(\mathbf{x}) + \mathbf{h}^\top\mathbf{c} + \tfrac{1}{2}\mathbf{h}^\top\mathbf{B}\,\mathbf{h}\,, \tag{2.14}$$

where $\mathbf{c}\in\mathbb{R}^n$ and the matrix $\mathbf{B}\in\mathbb{R}^{n\times n}$ is symmetric. The basic ideas of this section may be generalized to other forms of the model, but in this booklet we only need the form of L given in (2.14). Typically, the model is a second order Taylor expansion of F around $\mathbf{x}$, like the first three terms in the right-hand side of (1.4a), or $L(\mathbf{h})$ may be an approximation to this expansion. It is generally true that such a model is good only when $\mathbf{h}$ is sufficiently small. We shall introduce two methods that include this aspect in the determination of a step $\mathbf{h}$, which is a descent direction and which can be used with $\alpha=1$ in Algorithm 2.4.

In a *trust region method* we assume that we know a positive number Δ such that the model is sufficiently accurate inside a ball with radius Δ, centered at $\mathbf{x}$, and determine the step as

$$\mathbf{h} \;=\; \mathbf{h}_{\text{tr}} \;\equiv\; \text{argmin}_{\|\mathbf{h}\|\le\Delta}\{L(\mathbf{h})\}. \tag{2.15}$$

In a *damped method* the step is determined as

$$\mathbf{h} \;=\; \mathbf{h}_{\text{dm}} \;\equiv\; \text{argmin}_{\mathbf{h}}\{L(\mathbf{h}) + \tfrac{1}{2}\,\mu\,\mathbf{h}^\top\mathbf{h}\}, \tag{2.16}$$

where the *damping parameter* $\mu\ge 0$. The term $\frac{1}{2}\,\mu\,\mathbf{h}^\top\mathbf{h} = \frac{1}{2}\,\mu\,\|\mathbf{h}\|^2$ is seen to penalize large steps.

The central part of Algorithm 2.4 based on one of these methods has the form

```
Compute h by (2.15) or (2.16)
if  F(x+h) < F(x)
   x := x + h
Update Δ or μ
```
(2.17)

This corresponds to $\alpha=1$ if the step $\mathbf{h}$ satisfies the descending condition (2.1). Otherwise, $\alpha=0$, ie we do not move.[2] However, we are not stuck

[2] There are versions of these methods that include a proper line search to find a point $\mathbf{x}+\alpha\mathbf{h}$ with smaller F-value, and information gathered during the line search is used in the updating of Δ or μ. For many problems such versions use fewer iteration steps but a larger accumulated number of function values.

i.e. larger accumulated error

at $\mathbf{x}$ (unless $\mathbf{x} = \mathbf{x}^*$): by a proper modification of Δ or μ we aim at having better luck in the next iteration step.

Since $L(\mathbf{h})$ is assumed to be a good approximation to $F(\mathbf{x}+\mathbf{h})$ for $\mathbf{h}$ sufficiently small, the reason why the step failed is that $\mathbf{h}$ was too large, and should be reduced. Further, if the step is accepted, it may be possible to use a larger step from the new iterate and thereby reduce the number of steps needed before we reach $\mathbf{x}^*$.

The quality of the model with the computed step can be evaluated by the so-called *gain ratio*

$$\varrho \;=\; \frac{F(\mathbf{x}) - F(\mathbf{x}+\mathbf{h})}{L(\mathbf{0}) - L(\mathbf{h})}\;, \tag{2.18}$$

ie the ratio between the actual and predicted decrease in function value. By construction the denominator is positive, and the numerator is negative if the step was not downhill – it was too large and should be reduced.

With a trust region method we monitor the step length by the size of the radius Δ. The following updating strategy is widely used,

if $\varrho < 0.25$
  $\Delta := \Delta/2$
elseif $\varrho > 0.75$
  $\Delta := \max\{\Delta, 3 * \|h\|\}$ (2.19)

gain ratio较低，说明这个 approximation 比较差，
应该减小步长，不至于达到太偏的地方；
gain ratio较高，说明这个approximation 比较好，
应该继续保持或增大步长，以快速达到 true local minima

Thus, if $\varrho < \frac{1}{4}$, we decide to use smaller steps, while $\varrho > \frac{3}{4}$ indicates that it may be possible to use larger steps. A trust region algorithm is not sensitive to minor changes in the thresholds 0.25 and 0.75, the divisor $p_1 = 2$ or the factor $p_2 = 3$, but it is important that the numbers p_1 and p_2 are chosen so that the Δ-values cannot oscillate.

即：应该增大的时候不会反而比原来更小；反之则反。

In a damped method a small value of ϱ indicates that we should increase the damping factor and thereby increase the penalty on large steps. A large value of ϱ indicates that $L(\mathbf{h})$ is a good approximation to $F(\mathbf{x}+\mathbf{h})$ for the computed $\mathbf{h}$, and the damping may be reduced. A widely used strategy is the following, which is similar to (2.19), and was was originally proposed by Marquardt (1963),

if $\varrho < 0.25$
  $\mu := \mu * 2$
elseif $\varrho > 0.75$
  $\mu := \mu/3$ (2.20)

Again, the method is not sensitive to minor changes in the thresholds 0.25 and 0.75 or the numbers $p_1 = 2$ and $p_2 = 3$, but it is important that the numbers p_1 and p_2 are chosen so that the μ-values cannot oscillate. Experience shows that the discontinuous changes across the thresholds 0.25 and 0.75 can give rise to a "flutter" (illustrated in Example 3.7 on page 27) that can slow down convergence, and we demonstrated in Nielsen (1999) that the following strategy in general outperforms (2.20),

if $\varrho > 0$
  $\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}; \quad \nu := 2$
else
  $\mu := \mu * \nu; \quad \nu := 2 * \nu$ (2.21)

The factor ν is initialized to $\nu = 2$. Notice that a series of consecutive failures results in rapidly increasing μ-values. The two updating formulas are illustrated below.

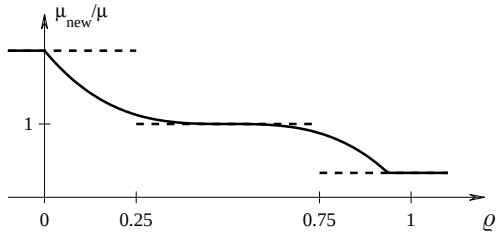

Figure 2.2. *Updating of μ by (2.21) with $\nu = 2$ (full line) Marquardt's strategy* (2.20) *(dasheded line).*

2.4.1. Computation of the step. In a damped method the step is computed as a stationary point for the function

$$\psi_\mu(\mathbf{h}) \;=\; L(\mathbf{h}) + \tfrac{1}{2}\,\mu\,\mathbf{h}^\top\mathbf{h}\;,$$

This means that $\mathbf{h}_{\text{dm}}$ is a solution to

$$\psi'_\mu(\mathbf{h}) \;=\; \mathbf{L}'(\mathbf{h}) + \mu\mathbf{h} \;=\; \mathbf{0}\;,$$

and from the definition of $L(\mathbf{h})$ in (2.14) we see that this is equivalent to

$$(\mathbf{B} + \mu\mathbf{I})\mathbf{h}_{\text{dm}} \;=\; -\mathbf{c}\;, \tag{2.22}$$

where $\mathbf{I}$ is the identity matrix. If μ is sufficiently large, the symmetric matrix $\mathbf{B}+\mu\mathbf{I}$ is positive definite (shown in Appendix A), and then it follows from Theorem 1.8 that $\mathbf{h}_{\text{dm}}$ is a minimizer for L.

Example 2.1. In a *damped Newton method* the model $L(\mathbf{h})$ is given by $\mathbf{c} = \mathbf{F}'(\mathbf{x})$ and $\mathbf{B} = \mathbf{F}''(\mathbf{x})$, and (2.22) takes the form

$$(\mathbf{F}''(\mathbf{x}) + \mu\mathbf{I})\mathbf{h}_{\text{dn}} \;=\; -\mathbf{F}'(\mathbf{x})\;.$$

$\mathbf{h}_{\text{dn}}$ is the so-called *damped Newton step*. If μ is very large, then

$$\mathbf{h}_{\text{dn}} \;\simeq\; -\frac{1}{\mu}\,\mathbf{F}'(\mathbf{x})\;,$$

ie a short step in a direction close to the steepest descent direction. On the other hand, if μ is very small, then $\mathbf{h}_{\text{dn}}$ is close to the Newton step $\mathbf{h}_{\text{n}}$. Thus, we can think of the damped Newton method as a hybrid between the steepest descent method and the Newton method. ∎

We return to damped methods in Section 3.2.

In a trust region method the step $\mathbf{h}_{\text{tr}}$ is the solution to a *constrained optimization problem*,

$$\begin{aligned} &\text{minimize} \quad L(\mathbf{h}) \\ &\text{subject to} \quad \mathbf{h}^\top\mathbf{h} \le \Delta^2\;. \end{aligned} \tag{2.23}$$

It is outside the scope of this booklet to discuss this problem in any detail (see Madsen et al (2004) or Section 4.1 in Nocedal and Wright (1999). We just want to mention a few properties.

If the matrix $\mathbf{B}$ in (2.14) is positive definite, then the unconstrained minimizer of L is the solution to

$$\mathbf{B}\mathbf{h} \;=\; -\mathbf{c}\;,$$

and if this is sufficiently small (if it satisfies $\mathbf{h}^\top\mathbf{h} \le \Delta^2$), then this is the desired step, $\mathbf{h}_{\text{tr}}$. Otherwise, the constraint is active, and the problem is more complicated. With a similar argument as we used on page 11, we can see that we do not have to compute the true solution to (2.23), and in Sections 3.3 and 3.4 we present two ways of computing an approximation to $\mathbf{h}_{\text{tr}}$.

Finally, we present two similarities between a damped method and a trust region method in the case where $\mathbf{B}$ is positive definite: In case the unconstrained minimizer is outside the trust region, it can be shown (Theorem 2.11 in Madsen et al (2004)) that there exists a $\lambda > 0$ such that

$$\mathbf{B}\mathbf{h}_{\text{tr}} + \mathbf{c} \;=\; -\lambda\mathbf{h}_{\text{tr}}\;. \tag{2.24a}$$

By reordering this equation and comparing it with (2.22) we see that $\mathbf{h}_{\text{tr}}$ is identical with the damped step $\mathbf{h}_{\text{dm}}$ computed with the damping parameter $\mu = \lambda$. On the other hand, one can also show (Theorem 5.11 in Frandsen et al (2004)) that if we compute $\mathbf{h}_{\text{dm}}$ for a given $\mu \ge 0$, then

$$\mathbf{h}_{\text{dm}} \;=\; \text{argmin}_{\|h\| \le \|\mathbf{h}_{\text{dm}}\|}\{L(\mathbf{h})\}\;, \tag{2.24b}$$

ie $\mathbf{h}_{\text{dm}}$ is equal to $\mathbf{h}_{\text{tr}}$ corresponding to the trust region radius $\Delta = \|\mathbf{h}_{\text{dm}}\|$. Thus, the two classes of methods are closely related, but there is not a simple formula for the connection between the Δ- and μ-values that give the same step.

This formula says: given a single row of J(x)^T, denoted with index j, multiplies the whole matrix f(x).
The \Sigma sign denotes the multiplication of each row of f(x) with the given row of J(x)^T, and the result is a single row of F'(x).
Therefore, we have derived the equation 3.4a.

3. NON-LINEAR LEAST SQUARES PROBLEMS

我的思考:
自己还没想明白

In the remainder of this lecture note we shall discuss methods for nonlinear least squares problems. Given a vector function $\mathbf{f} : \mathbf{R}^n \mapsto \mathbf{R}^m$ with $m \geq n$. We want to minimize $\|\mathbf{f}(\mathbf{x})\|$, or equivalently to find

why vector function ?

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x}}\{F(\mathbf{x})\} , \tag{3.1a}$$

where

$$F(\mathbf{x}) = \tfrac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 = \tfrac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2 = \tfrac{1}{2}\mathbf{f}(\mathbf{x})^\top\mathbf{f}(\mathbf{x}) . \tag{3.1b}$$

Least squares problems can be solved by general optimization methods, but we shall present special methods that are more efficient. In many cases they achieve better than linear convergence, sometimes even quadratic convergence, even though they do not need implementation of second derivatives.

In the description of the methods in this chapter we shall need formulas for derivatives of F: Provided that $\mathbf{f}$ has continuous second partial derivatives, we can write its *Taylor expansion* as

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) \;=\; \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2) , \tag{3.2a}$$

where $\mathbf{J} \in \mathbf{R}^{m\times n}$ is the *Jacobian*. This is a matrix containing the first partial derivatives of the function components, Jacobian is the matrix extension of the gradient.

Each row in the Jacobian is the transpose of the gradient of a single component of the output f.

$$(\mathbf{J}(\mathbf{x}))_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}) \tag{3.2b}$$

As regards $F : \mathbf{R}^n \mapsto \mathbf{R}$, it follows from the first formulation in (3.1b),

F(x)一直是 scalar-valued function.
f(x) 可能是 scalar-valued 也可能是 vector-valued.

that[1)]

$$\frac{\partial F}{\partial x_j}(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})\frac{\partial f_i}{\partial x_j}(\mathbf{x}) . \tag{3.3}$$

Thus, the gradient (1.4b) is

F'(x) : n x n matrix

$$\mathbf{F}'(\mathbf{x}) = \mathbf{J}(\mathbf{x})^\top\mathbf{f}(\mathbf{x}) . \tag{3.4a}$$

We shall also need the Hessian of F. From (3.3) we see that the element in position (j,k) is

$$\frac{\partial^2 F}{\partial x_j \partial x_k}(\mathbf{x}) \;=\; \sum_{i=1}^{m}\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x})\frac{\partial f_i}{\partial x_k}(\mathbf{x}) + f_i(\mathbf{x})\frac{\partial^2 f_i}{\partial x_j \partial x_k}(\mathbf{x})\right) ,$$

showing that

$$\mathbf{F}''(\mathbf{x}) \;=\; \mathbf{J}(\mathbf{x})^\top\mathbf{J}(\mathbf{x}) + \sum_{i=1}^{m} f_i(x)\mathbf{f}_i''(\mathbf{x}) . \tag{3.4b}$$

Example 3.1. The simplest case of (3.1) is when $\mathbf{f}(\mathbf{x})$ has the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{x} ,$$

where the vector $\mathbf{b} \in \mathbf{R}^m$ and matrix $\mathbf{A} \in \mathbf{R}^{m\times n}$ are given. We say that this is a *linear least squares problem*. In this case $\mathbf{J}(\mathbf{x}) = -\mathbf{A}$ for all $\mathbf{x}$, and from (3.4a) we see that In this case, f(x) is called the affine function, aka linear plus a constant term of x.

$$\mathbf{F}'(\mathbf{x}) = -\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}) .$$

This is zero for $\mathbf{x}^*$ determined as the solution to the so-called *normal equations*,

see a certain ppt lecture.

$$(\mathbf{A}^\top\mathbf{A})\mathbf{x}^* \;= \mathbf{A}^\top\mathbf{b} . \tag{3.5}$$

The problem can be written in the form

$$\mathbf{A}\mathbf{x}^* \simeq \mathbf{b} ,$$

and alternatively we can solve it via *orthogonal transformation*: Find an orthogonal matrix $\mathbf{Q}$ such that Q: m x m

[1)] If we had not used the factor $\frac{1}{2}$ in the definition (3.1b), we would have got an annoying factor of 2 in a lot of expressions.

$$\mathbf{Q}^\top \mathbf{A} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} ,$$

0 matrix here is ((m - n) x n) size.

where $\mathbf{R} \in \mathbf{R}^{n\times n}$ is upper triangular. The solution is found by back substitution in the system[2)]

$$\mathbf{R}\mathbf{x}^* = (\mathbf{Q}^\top \mathbf{b})_{1:n} .$$

Q^T * b is (m x m) * (m x 1) = (m x 1), a row vector.

This method is more accurate than the solution via the normal equations.

In MATLAB suppose that the arrays A and b hold the matrix $\mathbf{A}$ and vector $\mathbf{b}$, respectively. Then the command A\b returns the least squares solution computed via orthogonal transformation.

means may be approximative solution ?

As the title of the booklet suggests, we assume that $\mathbf{f}$ is nonlinear, and shall not discuss linear problems in detail. We refer to Chapter 2 in Madsen and Nielsen (2002) or Section 5.2 in Golub and Van Loan (1996). ∎

Example 3.2. In Example 1.1 we saw a nonlinear least squares problem arising from data fitting. Another application is in the solution of nonlinear systems of equations,

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{0} , \quad \text{where} \quad \mathbf{f} : \mathbf{R}^n \mapsto \mathbf{R}^n .$$

We can use *Newton-Raphson's method:* From an initial guess $\mathbf{x}_0$ we compute $\mathbf{x}_1, \mathbf{x}_2, \ldots$ by the following algorithm, which is based on seeking $\mathbf{h}$ so that $\mathbf{f}(\mathbf{x}+\mathbf{h}) = \mathbf{0}$ and ignoring the term $O(\|\mathbf{h}\|^2)$ in (3.2a),

$$\begin{aligned} &\text{Solve } \mathbf{J}(\mathbf{x}_k)\mathbf{h}_k = -\mathbf{f}(\mathbf{x}_k) \text{ for } \mathbf{h}_k \\ &\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k \ . \end{aligned} \tag{3.6}$$

Here, the Jacobian $\mathbf{J}$ is given by (3.2b). If $\mathbf{J}(\mathbf{x}^*)$ is nonsingular, then the method has quadratic final convergence, ie if $d_k = \|\mathbf{x}_k - \mathbf{x}^*\|$ is small, then $\|\mathbf{x}_{k+1} - \mathbf{x}^*\| = O(d_k^2)$. However, if $\mathbf{x}_k$ is far from $\mathbf{x}^*$, then we risk to get even further away.

We can reformulate the problem in a way that enables us to use all the "tools" that we are going to present in this chapter: A solution of (3.6) is a global minimizer of the function F defined by (3.1),

$$F(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2 ,$$

[2)] An expression like $\mathbf{u}_{p:q}$ is used to denote the subvector with elements $u_i,\ i = p, \ldots, q$. The ith row and jth column of a matrix $\mathbf{A}$ is denoted $\mathbf{A}_{i,:}$ and $\mathbf{A}_{:,j}$, respectively.

colon ":" denotes "all", i.e. all elements along a given row or col.

since $F(\mathbf{x}^*) = 0$ and $F(\mathbf{x}) > 0$ if $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. We may eg replace the updating of the approximate solution in (3.6) by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{h}_k \ ,$$

where α_k is found by line search applied to the function $\varphi(\alpha) = F(\mathbf{x}_k + \alpha\mathbf{h}_k)$.

As a specific example we shall consider the following problem, taken from Powell (1970),

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1 \\ \frac{10x_1}{x_1+0.1} + 2x_2^2 \end{bmatrix} ,$$

with $\mathbf{x}^* = \mathbf{0}$ as the only solution. The Jacobian is

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ (x_1+0.1)^{-2} & 4x_2 \end{bmatrix} ,$$

which is singular at the solution.

If we take $\mathbf{x}_0 = [3,\ 1]^\top$ and use the above algorithm with exact line search, then the iterates converge to $\mathbf{x}_c \simeq [1.8016,\ 0]^\top$, which is **not** a solution. On the other hand, it is easily seen that the iterates given by Algorithm (3.6) are $\mathbf{x}_k = [0, y_k]^\top$ with $y_{k+1} = \frac{1}{2}y_k$, ie we have linear convergence to the solution.

In a number of examples we shall return to this problem to see how different methods handle it. ∎

3.1. The Gauss–Newton Method

This method is the basis of the very efficient methods we will describe in the next sections. It is based on implemented first derivatives of the components of the vector function. In special cases it can give quadratic convergence as the Newton-method does for general optimization, see Frandsen et al (2004).

The Gauss–Newton method is based on a linear approximation to the components of $\mathbf{f}$ (a *linear model* of $\mathbf{f}$) in the neighbourhood of $\mathbf{x}$: For small $\|\mathbf{h}\|$ we see from the Taylor expansion (3.2) that

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) \ \simeq \ \ell(\mathbf{h}) \ \equiv \ \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} \ . \tag{3.7a}$$

Inserting this in the definition (3.1) of F we see that

$$F(\mathbf{x}+\mathbf{h}) \simeq L(\mathbf{h}) \equiv \tfrac{1}{2}\boldsymbol{\ell}(\mathbf{h})^\top \boldsymbol{\ell}(\mathbf{h})$$
$$= \tfrac{1}{2}\mathbf{f}^\top\mathbf{f} + \mathbf{h}^\top\mathbf{J}^\top\mathbf{f} + \tfrac{1}{2}\mathbf{h}^\top\mathbf{J}^\top\mathbf{J}\mathbf{h}$$
$$= F(\mathbf{x}) + \mathbf{h}^\top\mathbf{J}^\top\mathbf{f} + \tfrac{1}{2}\mathbf{h}^\top\mathbf{J}^\top\mathbf{J}\mathbf{h} \qquad (3.7b)$$

(with $\mathbf{f} = \mathbf{f}(\mathbf{x})$ and $\mathbf{J} = \mathbf{J}(\mathbf{x})$). The *Gauss–Newton step* $\mathbf{h}_{\text{gn}}$ minimizes $L(\mathbf{h})$,

$$\mathbf{h}_{\text{gn}} = \operatorname{argmin}_{\mathbf{h}}\{L(\mathbf{h})\} .$$

It is easily seen that the gradient and the Hessian of L are

$$\mathbf{L}'(\mathbf{h}) = \mathbf{J}^\top\mathbf{f} + \mathbf{J}^\top\mathbf{J}\mathbf{h}, \qquad \mathbf{L}''(\mathbf{h}) = \mathbf{J}^\top\mathbf{J} . \qquad (3.8)$$

Comparison with (3.4a) shows that $\mathbf{L}'(\mathbf{0}) = \mathbf{F}'(\mathbf{x})$. Further, we see that the matrix $\mathbf{L}''(\mathbf{h})$ is independent of $\mathbf{h}$. It is symmetric and if $\mathbf{J}$ has *full rank*, ie if the columns are linearly independent, then $\mathbf{L}''(\mathbf{h})$ is also positive definite, cf Appendix A. This implies that $L(\mathbf{h})$ has a unique minimizer, which can be found by solving

cf. Theorem 1.8
But we haven't proved that h is the stationary point.

$$(\mathbf{J}^\top\mathbf{J})\mathbf{h}_{\text{gn}} = -\mathbf{J}^\top\mathbf{f} . \qquad (3.9)$$

by setting L'(h) to zero.

This is a descent direction for F since

$$\mathbf{h}_{\text{gn}}{}^\top\mathbf{F}'(\mathbf{x}) = \mathbf{h}_{\text{gn}}{}^\top(\mathbf{J}^\top\mathbf{f}) = -\mathbf{h}_{\text{gn}}{}^\top(\mathbf{J}^\top\mathbf{J})\mathbf{h}_{\text{gn}} < 0 . \qquad (3.10)$$

By substituting eq 3.9 into Definition 2.6

Thus, we can use $\mathbf{h}_{\text{gn}}$ for $\mathbf{h}_{\text{d}}$ in Algorithm 2.4. The typical step is

$$\begin{aligned} &\text{Solve } (\mathbf{J}^\top\mathbf{J})\mathbf{h}_{\text{gn}} = -\mathbf{J}^\top\mathbf{f} \\ &\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_{\text{gn}} \end{aligned} \qquad (3.11)$$

where α is found by line search. The classical Gauss-Newton method uses $\alpha = 1$ in all steps. The method with line search can be shown to have guaranteed convergence, provided that

a) $\{\mathbf{x} \mid F(\mathbf{x}) \le F(\mathbf{x}_0)\}$ is bounded, and

b) the Jacobian $\mathbf{J}(\mathbf{x})$ has full rank in all steps.

In chapter 2 we saw that Newton's method for optimization has quadratic convergence. This is normally not the case with the Gauss-Newton method.

To see this, we compare the search directions used in the two methods,

$$\mathbf{F}''(\mathbf{x})\mathbf{h}_{\text{n}} = -\mathbf{F}'(\mathbf{x}) \quad \text{and} \quad \mathbf{L}''(\mathbf{h})\mathbf{h}_{\text{gn}} = -\mathbf{L}'(\mathbf{0}) .$$

We already remarked at (3.8) that the two right-hand sides are identical, but from (3.4b) and (3.8) we see that the coefficient matrices differ:

$$\mathbf{F}''(\mathbf{x}) = \mathbf{L}''(\mathbf{h}) + \sum_{i=1}^{m} f_i(x)\mathbf{f}_i''(\mathbf{x}) . \qquad (3.12)$$

Therefore, if $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, then $\mathbf{L}''(\mathbf{h}) \simeq \mathbf{F}''(\mathbf{x})$ for $\mathbf{x}$ close to $\mathbf{x}^*$, and we get quadratic convergence also with the Gauss-Newton method. We can expect superlinear convergence if the functions $\{f_i\}$ have small curvatures or if the $\{|f_i(\mathbf{x}^*)|\}$ are small, but in general we must expect linear convergence. It is remarkable that the value of $F(\mathbf{x}^*)$ controls the convergence speed.

Example 3.3. Consider the simple problem with $n = 1$, $m = 2$ given by

$$\mathbf{f}(x) = \begin{bmatrix} x+1 \\ \lambda x^2 + x - 1 \end{bmatrix} . \quad F(x) = \tfrac{1}{2}(x+1)^2 + \tfrac{1}{2}(\lambda x^2 + x - 1)^2 .$$

It follows that

$$F'(x) = 2\lambda^2 x^3 + 3\lambda x^2 - 2(\lambda - 1)x ,$$

so $x = 0$ is a stationary point for F. Now,

$$F''(x) = 6\lambda^2 x^2 + 6\lambda x - 2(\lambda - 1) .$$

This shows that if $\lambda < 1$, then $\mathbf{F}''(0) > 0$, so $x = 0$ is a local minimizer – actually, it is the global minimizer.

The Jacobian is

$$\mathbf{J}(x) = \begin{bmatrix} 1 \\ 2\lambda x + 1 \end{bmatrix} ,$$

and the classical Gauss-Newton method from x_k gives

$$x_{k+1} = x_k - \frac{2\lambda^2 x_k^3 + 3\lambda x_k^2 - 2(\lambda - 1)x_k}{2 + 4\lambda x_k + 4\lambda^2 x_k^2} .$$

from eq 3.11

Now, if $\lambda \neq 0$ and x_k is close to zero, then

$$x_{k+1} = x_k + (\lambda - 1)x_k + O(x_k^2) = \lambda x_k + O(x_k^2) .$$

怎么算的?
为什么 一定是 二阶 无穷小?

Note previously, we have set that \lambda < 1.
This means, the x_k sequence won't converge.

Thus, if $|\lambda| < 1$, we have linear convergence. If $\lambda < -1$, then the classical Gauss-Newton method cannot find the minimizer. Eg with $\lambda = -2$ and $x_0 = 0.1$ we get a seemingly chaotic behaviour of the iterates,

k	x_k
0	0.1000
1	−0.3029
2	0.1368
3	−0.4680
⋮	⋮

Finally, if $\lambda = 0$, then

$$x_{k+1} = x_k - x_k = 0 \,,$$

ie we find the solution in one step. The reason is that in this case $\mathbf{f}$ is a linear function. ∎

Example 3.4. For the data fitting problem from Example 1.1 the ith row of the Jacobian matrix is

$$\mathbf{J}(\mathbf{x})_{i,:} = \begin{bmatrix} -x_3 t_i e^{x_1 t_i} & -x_4 t_i e^{x_2 t_i} & -e^{x_1 t_i} & -e^{x_2 t_i} \end{bmatrix} .$$

If the problem is *consistent* (ie $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$), then the Gauss-Newton method with line search will have quadratic final convergence, provided that x_1^* is significantly different from x_2^*. If $x_1^* = x_2^*$, then $\operatorname{rank}(\mathbf{J}(\mathbf{x}^*)) \le 2$, and the Gauss-Newton method fails.

If one or more measurement errors are large, then $\mathbf{f}(\mathbf{x}^*)$ has some large components, and this may slow down the convergence.

In MATLAB we can give a very compact function for computing $\mathbf{f}$ and $\mathbf{J}$: Suppose that `x` holds the current iterate and that the $m \times 2$ array `ty` holds the coordinates of the data points. The following function returns `f` and `J` containing $\mathbf{f}(\mathbf{x})$ and $\mathbf{J}(\mathbf{x})$, respectively.

```
function  [f, J] = fitexp(x, ty)
  t = ty(:,1); y = ty(:,2);
  E = exp(t * [x(1), x(2)]);
  f = y - E*[x(3); x(4)];
  J = -[x(3)*t.*E(:,1), x(4)*t.*E(:,2), E];
```

∎

Example 3.5. Consider the problem from Example 3.2, $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$ with $\mathbf{f} : \mathbf{R}^n \mapsto \mathbf{R}^n$. If we use Newton-Raphson's method to solve this problem, the typical iteration step is

$$\text{Solve } \mathbf{J}(\mathbf{x})\mathbf{h}_{\text{nr}} = -\mathbf{f}(\mathbf{x}); \quad \mathbf{x} := \mathbf{x} + \mathbf{h}_{\text{nr}} \,.$$

The Gauss-Newton method applied to the minimization of $F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$ has the typical step

$$\text{Solve } (\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}))\mathbf{h}_{\text{gn}} = -\mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}); \quad \mathbf{x} := \mathbf{x} + \mathbf{h}_{\text{gn}} \,.$$

Note, that $\mathbf{J}(\mathbf{x})$ is a square matrix, and we assume that it is nonsingular. Then $(\mathbf{J}(\mathbf{x})^\top)^{-1}$ exists, and it follows that $\mathbf{h}_{\text{gn}} = \mathbf{h}_{\text{nr}}$. Therefore, when applied to Powell's problem from Example 3.2, the Gauss-Newton method will have the same troubles as discussed for Newton-Raphson's method in that example. ∎

These examples show that the Gauss-Newton method may fail, both with and without a line search. Still, in many applications it gives quite good performance, though it normally only has linear convergence as opposed to the quadratic convergence from Newton's method with implemented second derivatives.

In Sections 3.2 and 3.3 we give two methods with superior global performance, and in Section 3.4 we give modifications to the first method so that we achieve superlinear final convergence.

the damped version of the Gauss-Newton method.

3.2. The Levenberg–Marquardt Method

Levenberg (1944) and later Marquardt (1963) suggested to use a *damped Gauss-Newton method*, cf Section 2.4. The step $\mathbf{h}_{\text{lm}}$ is defined by the following modification to (3.9),

$$(\mathbf{J}^\top \mathbf{J} + \mu \mathbf{I})\mathbf{h}_{\text{lm}} = -\mathbf{g} \quad \text{with } \mathbf{g} = \mathbf{J}^\top \mathbf{f} \text{ and } \mu \ge 0 \,. \tag{3.13}$$

Here, $\mathbf{J} = \mathbf{J}(\mathbf{x})$ and $\mathbf{f} = \mathbf{f}(\mathbf{x})$. The damping parameter μ has several effects:

a) For all $\mu > 0$ the coefficient matrix is positive definite, and this ensures that $\mathbf{h}_{\text{lm}}$ is a descent direction, cf (3.10).

The matrix on the left-hand side of h.

Because the coefficient matrix is full column rank.

For large values of \mu, the only significant part of the coefficient matrix is the \mu * I with the J^T * J dropped.

b) For large values of μ we get

cf. Page 9, Example 2.1

$$\mathbf{h}_{\text{lm}} \simeq -\frac{1}{\mu}\,\mathbf{g} = -\frac{1}{\mu}\,\mathbf{F}'(\mathbf{x}) ,$$

cf. eq 2.8

ie a short step in the steepest descent direction. This is good if the current iterate is far from the solution.

Thus L-M method is a hybrid method.

c) If μ is very small, then $\mathbf{h}_{\text{lm}} \simeq \mathbf{h}_{\text{gn}}$, which is a good step in the final stages of the iteration, when $\mathbf{x}$ is close to $\mathbf{x}^*$. If $F(\mathbf{x}^*) = 0$ (or very small), then we can get (almost) quadratic final convergence.

Thus, the damping parameter influences both the direction and the size of the step, and this leads us to make a method **without** a specific line search. The choice of initial μ-value should be related to the size of the elements in $\mathbf{A}_0 = \mathbf{J}(\mathbf{x_0})^\top \mathbf{J}(\mathbf{x_0})$, eg by letting

$$\mu_0 = \tau \cdot \max_i\{a_{ii}^{(0)}\} , \qquad \text{(3.14)}$$

superscript zero means x_0 ?

where τ is chosen by the user.[3] During iteration the size of μ can be updated as described in Section 2.4. The updating is controlled by the *gain ratio*

$$\varrho = \frac{F(\mathbf{x}) - F(\mathbf{x}+\mathbf{h}_{\text{lm}})}{L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}})} ,$$

where the denominator is the gain predicted by the linear model (3.7b),

$$\begin{aligned} L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}}) &= -\mathbf{h}_{\text{lm}}{}^\top \mathbf{J}^\top \mathbf{f} - \tfrac{1}{2}\mathbf{h}_{\text{lm}}{}^\top \mathbf{J}^\top \mathbf{J}\mathbf{h}_{\text{lm}} \\ &= -\tfrac{1}{2}\mathbf{h}_{\text{lm}}{}^\top\big(\mathbf{2g} + (\mathbf{J}^\top\mathbf{J} + \mu\mathbf{I} - \mu\mathbf{I})\mathbf{h}_{\text{lm}}\big) \\ &= \tfrac{1}{2}\mathbf{h}_{\text{lm}}{}^\top(\mu\mathbf{h}_{\text{lm}} - \mathbf{g}) \ . \end{aligned}$$

Note that both $\mathbf{h}_{\text{lm}}{}^\top\mathbf{h}_{\text{lm}}$ and $-\mathbf{h}_{\text{lm}}{}^\top\mathbf{g}$ are positive, so $L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}})$ is guaranteed to be positive.

A large value of ϱ indicates that $L(\mathbf{h}_{\text{lm}})$ is a good approximation to $F(\mathbf{x}+\mathbf{h}_{\text{lm}})$, and we can decrease μ so that the next Levenberg-Marquardt

[3] The algorithm is not very sensitive to the choice of τ, but as a rule of thumb, one should use a small value, eg $\tau = 10^{-6}$ if $\mathbf{x}_0$ is believed to be a good approximation to $\mathbf{x}^*$. Otherwise, use $\tau = 10^{-3}$ or even $\tau = 1$.

According to the hybrid property, the rate of convergence of the L-M method is: quadratic convergence if F (x*) = 0, linear convergence otherwise

step is closer to the Gauss-Newton step. If ϱ is small (maybe even negative), then $L(\mathbf{h}_{\text{lm}})$ is a poor approximation, and we should increase μ with the twofold aim of getting closer to the steepest descent direction **and** reducing the step length. These goals can be met in different ways, see page 14 and Example 3.7 below.

The *stopping criteria* for the algorithm should reflect that at a global minimizer we have $\mathbf{F}'(\mathbf{x}^*) = \mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, so we can use

$$\|\mathbf{g}\|_\infty \le \varepsilon_1 , \qquad \text{(3.15a)}$$

where ε_1 is a small, positive number, chosen by the user. Another relevant criterion is to stop if the change in $\mathbf{x}$ is small,

$$\|\mathbf{x}_{\text{new}} - \mathbf{x}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2) . \qquad \text{(3.15b)}$$

This expression gives a gradual change from relative step size ε_2 when $\|\mathbf{x}\|$ is large to absolute step size ε_2^2 if $\mathbf{x}$ is close to $\mathbf{0}$. Finally, as in all iterative processes we need a safeguard against an infinite loop,

$$k \ge k_{\max} . \qquad \text{(3.15c)}$$

Also ε_2 and $k_{\max}$ are chosen by the user.

The last two criteria come into effect eg if ε_1 is chosen so small that effects of rounding errors have large influence. This will typically reveal itself in a poor accordance between the actual gain in F and the gain predicted by the linear model (3.7b), and will result in μ being augmented in every step. The strategy (2.21) for augmenting μ implies that in this case μ grows fast, resulting in small $\|\mathbf{h}_{\text{lm}}\|$, and the process will be stopped by (3.15b).

The algorithm is summarized below.

Example 3.6. By comparing (3.9) and the normal equations (3.5) we see that $\mathbf{h}_{\text{gn}}$ is simply the least squares solution to the linear problem

$$\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} \simeq \mathbf{0} .$$

Similarly, the L-M equations (3.13) are the normal equations for the linear problem

Algorithm 3.16. Levenberg–Marquardt method

begin
$k := 0; \quad \nu := 2; \quad \mathbf{x} := \mathbf{x}_0$
$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$
found $:= (\|\mathbf{g}\|_\infty \le \varepsilon_1); \quad \mu := \tau * \max\{a_{ii}\}$
while (**not** *found*) **and** $(k < k_{\max})$
$k := k+1; \quad$ Solve $(\mathbf{A} + \mu\mathbf{I})\mathbf{h}_{\text{lm}} = -\mathbf{g}$
if $\|\mathbf{h}_{\text{lm}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$
found $:=$ **true**
else
$\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{lm}}$
$\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\text{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}}))$
if $\varrho > 0$ {step acceptable}
$\mathbf{x} := \mathbf{x}_{\text{new}}$
$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$
found $:= (\|\mathbf{g}\|_\infty \le \varepsilon_1)$
$\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}; \quad \nu := 2$
else
$\mu := \mu * \nu; \quad \nu := 2 * \nu$
end

$$\begin{bmatrix} \mathbf{f}(\mathbf{x}) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{J}(\mathbf{x}) \\ \sqrt{\mu}\mathbf{I} \end{bmatrix} \mathbf{h} \simeq \mathbf{0}\,.$$

As mentioned in Example 3.1, the most accurate solution is found via orthogonal transformation. However, the solution $\mathbf{h}_{\text{lm}}$ is just a step in an iterative process, and needs not be computed very accurately, and since the solution via the normal equations is "cheaper", this method is normally employed. ∎

Example 3.7. We have used Algorithm 3.16 on the data fitting problem from Examples 1.1 and 3.4. Figure 1.1 indicates that both x_1 and x_2 are negative and that $M(\mathbf{x}^*, 0) \simeq 0$. These conditions are satisfied by $\mathbf{x}_0 = [-1,\ -2,\ 1,\ -1]^\top$. Further, we used $\tau = 10^{-3}$ in the expression (3.14) for μ_0 and the stopping criteria

given by (3.15) with $\varepsilon_1 = \varepsilon_2 = 10^{-8}$, $k_{\max} = 200$. The algorithm stopped after 62 iteration steps with $\mathbf{x} \simeq [-4,\ -5,\ 4,\ -4]^\top$. The performance is illustrated below; note the logarithmic ordinate axis.

This problem is not consistent, so we could expect linear final convergence. The last 7 iteration steps indicate a much better (superlinear) convergence. The explanation is, that the $\mathbf{f}_i''(\mathbf{x})$ are slowly varying functions of t_i, and the $f_i(\mathbf{x}^*)$ have "random" sign, so that the contributions to the "forgotten term" in (3.12) almost cancel out. Such a situation occurs in many data fitting applications.

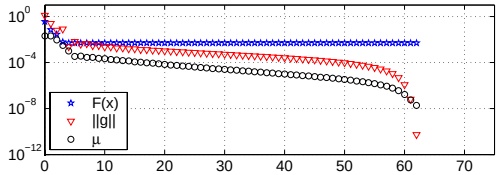

Figure 3.2a. *The L-M method applied to the fitting problem from Example 1.1.*

For comparison, Figure 3.2b shows the performance with the updating strategy (2.20). From step 5 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. This slows down the convergence, but the final stage is as in Figure 3.2a.

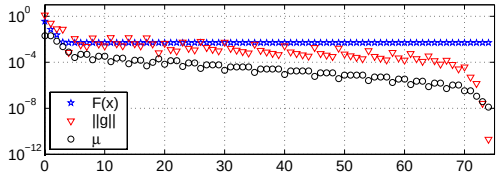

Figure 3.2b. *Performance with updating strategy (2.20).* ∎

Example 3.8. Figure 3.3 illustrates the performance of Algorithm 3.16 applied to Powell's problem from Examples 3.2 and 3.5. The starting point is $\mathbf{x}_0 = [3,\ 1]^\top$, μ_0 given by $\tau = 1$ in (3.14), and we use $\varepsilon_1 = \varepsilon_2 = 10^{-15}$, $k_{\max} = 100$ in the stopping criteria (3.15).

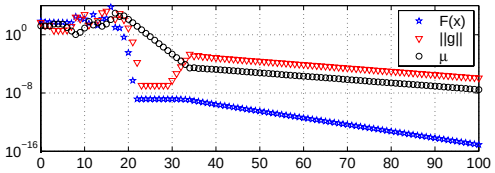

Figure 3.3. *The L-M method applied to Powell's problem.*

The iteration seems to stall between steps 22 and 30. This as an effect of the (almost) singular Jacobian matrix. After that there seems to be linear convergence. The iteration is stopped by the "safeguard" at the point $\mathbf{x} = [\,\texttt{-3.82e-08},\ \texttt{-1.38e-03}\,]^\top$. This is a better approximation to $\mathbf{x}^* = \mathbf{0}$ than we found in Example 3.2, but we want to be able to do even better; see Examples 3.10 and 3.17. ∎

the trust region version of the Gauss-Newton method.

3.3. Powell's Dog Leg Method

As the Levenberg–Marquardt method, this method works with combinations of the Gauss–Newton and the steepest descent directions. Now, however controlled explicitly via the radius of a *trust region*, cf Section 2.4. Powell's name is connected to the algorithm because he proposed how to find an approximation to $\mathbf{h}_{\text{tr}}$, defined by (2.23).

Given $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$. At the current iterate $\mathbf{x}$ the Gauss–Newton step $\mathbf{h}_{\text{gn}}$ is the least squares solution to the linear system

$$\mathbf{J}(\mathbf{x})\mathbf{h} \;\simeq\; -\mathbf{f}(\mathbf{x})\,. \tag{3.17}$$

It can be computed by solving the normal equations

$$\left(\mathbf{J}(\mathbf{x})^\top\mathbf{J}(\mathbf{x})\right)\mathbf{h}_{\text{gn}} \;=\; -\mathbf{J}(\mathbf{x})^\top\mathbf{f}(\mathbf{x})\,. \tag{3.18a}$$

The steepest descent direction is given by

$$\mathbf{h}_{\text{sd}} \;=\; -\mathbf{g} \;=\; -\mathbf{J}(\mathbf{x})^\top\mathbf{f}(\mathbf{x})\,. \tag{3.18b}$$

This is a direction, **not** a step, and to see how far we should go, we look at the linear model

$$\begin{aligned}
\mathbf{f}(\mathbf{x}+\alpha\mathbf{h}_{\text{sd}}) \;&\simeq\; \mathbf{f}(\mathbf{x}) + \alpha\mathbf{J}(\mathbf{x})\mathbf{h}_{\text{sd}} \\
\Downarrow \\
F(\mathbf{x}+\alpha\mathbf{h}_{\text{sd}}) \;&\simeq\; \tfrac{1}{2}\|\mathbf{f}(\mathbf{x}) + \alpha\mathbf{J}(\mathbf{x})\mathbf{h}_{\text{sd}}\|^2 \\
&=\; F(\mathbf{x}) + \alpha\mathbf{h}_{\text{sd}}{}^\top\mathbf{J}(\mathbf{x})^\top\mathbf{f}(\mathbf{x}) + \tfrac{1}{2}\alpha^2\|\mathbf{J}(\mathbf{x})\mathbf{h}_{\text{sd}}\|^2\,.
\end{aligned}$$

This function of α is minimal for

$$\alpha \;=\; -\,\frac{\mathbf{h}_{\text{sd}}{}^\top\mathbf{J}(\mathbf{x})^\top\mathbf{f}(\mathbf{x})}{\|\mathbf{J}(\mathbf{x})\mathbf{h}_{\text{sd}}\|^2} \;=\; \frac{\|\mathbf{g}\|^2}{\|\mathbf{J}(\mathbf{x})\mathbf{g}\|^2}\,. \tag{3.19}$$

Now we have two candidates for the step to take from the current point $\mathbf{x}$: $\mathbf{a} = \alpha\mathbf{h}_{\text{sd}}$ and $\mathbf{b} = \mathbf{h}_{\text{gn}}$. Powell suggested to use the following strategy for choosing the step, when the trust region has radius Δ. The last case in the strategy is illustrated in Figure 3.4.

$$\begin{aligned}
&\textbf{if}\ \ \|\mathbf{h}_{\text{gn}}\| \le \Delta \\
&\quad \mathbf{h}_{\text{dl}} := \mathbf{h}_{\text{gn}} \\
&\textbf{elseif}\ \ \|\alpha\mathbf{h}_{\text{sd}}\| \ge \Delta \\
&\quad \mathbf{h}_{\text{dl}} := (\Delta/\|\mathbf{h}_{\text{sd}}\|)\mathbf{h}_{\text{sd}} \\
&\textbf{else} \\
&\quad \mathbf{h}_{\text{dl}} := \alpha\mathbf{h}_{\text{sd}} + \beta(\mathbf{h}_{\text{gn}} - \alpha\mathbf{h}_{\text{sd}}) \\
&\qquad \text{with } \beta \text{ chosen so that } \|\mathbf{h}_{\text{dl}}\| = \Delta\,.
\end{aligned} \tag{3.20a}$$

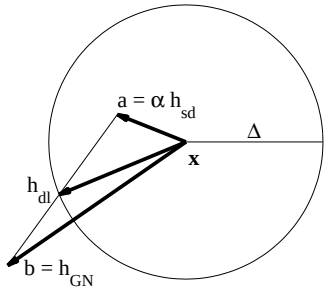

Figure 3.4. *Trust region and Dog Leg step.*[4)]

[4)] The name *Dog Leg* is taken from golf: The fairway at a "dog leg hole" has a shape as the line from $\mathbf{x}$ (the tee point) via the end point of $\mathbf{a}$ to the end point of $\mathbf{h}_{\text{dl}}$ (the hole). Powell is a keen golfer!

With $\mathbf{a}$ and $\mathbf{b}$ as defined above, and $c = \mathbf{a}^\top(\mathbf{b}-\mathbf{a})$ we can write

$$\psi(\beta) \equiv \|\mathbf{a}+\beta(\mathbf{b}-\mathbf{a})\|^2 - \Delta^2 = \|\mathbf{b}-\mathbf{a}\|^2\beta^2 + 2c\beta + \|\mathbf{a}\|^2 - \Delta^2 .$$

We seek a root for this second degree polynomial, and note that $\psi \to +\infty$ for $\beta \to -\infty$; $\psi(0) = \|\mathbf{a}\|^2 - \Delta^2 < 0$; $\psi(1) = \|\mathbf{h}_{\text{gn}}\|^2 - \Delta^2 > 0$. Thus, ψ has one negative root and one root in $]0,1[$. We seek the latter, and the most accurate computation of it is given by

$$\begin{aligned} &\textbf{if } c \le 0 \\ &\quad \beta = \left(-c + \sqrt{c^2 + \|\mathbf{b}-\mathbf{a}\|^2(\Delta^2 - \|\mathbf{a}\|^2)}\right) / \|\mathbf{b}-\mathbf{a}\|^2 \\ &\textbf{else} \\ &\quad \beta = \left(\Delta^2 - \|\mathbf{a}\|^2\right) / \left(c + \sqrt{c^2 + \|\mathbf{b}-\mathbf{a}\|^2(\Delta^2 - \|\mathbf{a}\|^2)}\right) \end{aligned} \tag{3.20b}$$

As in the L-M method we can use the gain ratio

$$\varrho = (F(\mathbf{x}) - F(\mathbf{x}+\mathbf{h}_{\text{dl}})) \,/ (L(\mathbf{0}) - L(\mathbf{h}_{\text{dl}}))$$

to monitor the iteration. Again, L is the linear model

$$L(\mathbf{h}) = \tfrac{1}{2}\|\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h}\|^2 .$$

In the L-M method we used ϱ to control the size of the damping parameter. Here, we use it to control the radius Δ of the trust region. A large value of ϱ indicates that the linear model is good. We can increase Δ and thereby take longer steps, and they will be closer to the Gauss-Newton direction. If ϱ is small (maybe even negative) then we reduce Δ, implying smaller steps, closer to the steepest descent direction. Below we summarize the algorithm.

We have the following remarks.

1° Initialization. $\mathbf{x}_0$ and Δ_0 should be supplied by the user.

2° We use the stopping criteria (3.15) supplemented with $\|\mathbf{f}(\mathbf{x})\|_\infty \le \varepsilon_3$, reflecting that $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$ in case of $m = n$, ie a nonlinear system of equations.

3° If $m = n$, then "$\simeq$" is replaced by "$=$", cf (3.6), and we do not use the detour around the normal equations (3.18a); see Example 3.9.

Algorithm 3.21. Dog Leg Method

begin
  $k := 0$; $\mathbf{x} := \mathbf{x}_0$; $\Delta := \Delta_0$; $\mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$ {1°}
  found := $(\|\mathbf{f}(\mathbf{x})\|_\infty \le \varepsilon_3)$ **or** $(\|\mathbf{g}\|_\infty \le \varepsilon_1)$ {2°}
  while (**not** *found*) **and** $(k < k_{\max})$
    $k := k+1$; Compute α by (3.19)
    $\mathbf{h}_{\text{sd}} := -\alpha\mathbf{g}$; Solve $\mathbf{J}(\mathbf{x})\mathbf{h}_{\text{gn}} \simeq -\mathbf{f}(\mathbf{x})$ {3°}
    Compute $\mathbf{h}_{\text{dl}}$ by (3.20)
    if $\|\mathbf{h}_{\text{dl}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$
      found := **true**
    else
      $\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{dl}}$
      $\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\text{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\text{dl}}))$ {4°}
      if $\varrho > 0$
        $\mathbf{x} := \mathbf{x}_{\text{new}}$; $\mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$
        found := $(\|\mathbf{f}(\mathbf{x})\|_\infty \le \varepsilon_3)$ **or** $(\|\mathbf{g}\|_\infty \le \varepsilon_1)$
      if $\varrho > 0.75$ {5°}
        $\Delta := \max\{\Delta, 3*\|\mathbf{h}_{\text{dl}}\|\}$
      elseif $\varrho < 0.25$
        $\Delta := \Delta/2$; *found* := $(\Delta \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2))$ {6°}
end

4° Corresponding to the three cases in (3.20a) we can show that

$$L(\mathbf{0}) - L(\mathbf{h}_{\text{dl}}) = \begin{cases} F(\mathbf{x}) & \text{if } \mathbf{h}_{\text{dl}} = \mathbf{h}_{\text{gn}} \\ \dfrac{\Delta(2\|\alpha\mathbf{g}\| - \Delta)}{2\alpha} & \text{if } \mathbf{h}_{\text{dl}} = \dfrac{-\Delta}{\|\mathbf{g}\|}\mathbf{g} \\ \tfrac{1}{2}\alpha(1-\beta)^2\|\mathbf{g}\|^2 + \beta(2-\beta)F(\mathbf{x}) & \text{otherwise} \end{cases}$$

5° Strategy (2.19) is used to update the trust region radius.

6° Extra stopping criterion. If $\Delta \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$, then (3.15b) will surely be satisfied in the next step.

Example 3.9. In Example 3.6 we briefly discussed the computation of the step $\mathbf{h}_{\text{lm}}$ and argued that we might as well compute it via the normal equations formulation (3.13). Provided that μ is not very small, the matrix is reasonably well conditioned, and there will be no excessive effects of rounding errors.

The Dog Leg method is intended perform well also on nonlinear systems of equations, ie where (3.17) is a square system of linear equations

$$\mathbf{J}(\mathbf{x})\mathbf{h} \;=\; -\mathbf{f}(\mathbf{x}) \;,$$

with the solution $\mathbf{h} = \mathbf{h}_{\text{nr}}$, the Newton-Raphson step, cf Example 3.2. The Jacobian $\mathbf{J}$ may be ill-conditioned (even singular), in which case rounding errors tend to dominate the solution. This problem is worsened if we use (3.18a) to compute $\mathbf{h}_{\text{gn}}$.

In the implementation `dogleg` in immoptibox the solution to (3.17) is computed with respect to these problems. If the columns of $\mathbf{J}(\mathbf{x})$ are not significantly linearly independent, then the least squares solution $\mathbf{h}$ is not unique, and $\mathbf{h}_{\text{gn}}$ is computed as the $\mathbf{h}$ with minimum norm. Some details of this computation are given in Appendix B. ∎

Example 3.10. Figure 3.5 illustrates the performance of the Dog Leg method applied to Powell's problem from Examples 3.2 and 3.8 with starting point $\mathbf{x}_0 \;=\; [\,3,\;1\,]^\top$, $\Delta_0 = 1$ and the stopping criteria given by $\varepsilon_1 = \varepsilon_2 = 10^{-15}$, $\varepsilon_3 = 10^{-20}$, $k_{\max} = 100$.

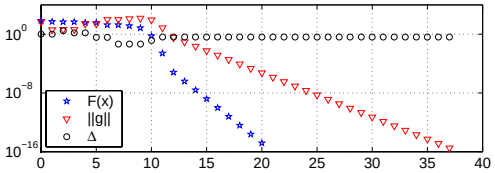

Figure 3.5. *Dog Leg method applied to Powell's problem.*

The iteration stopped after 37 steps because of a small gradient, and returned $\mathbf{x} = [\,-2.41{\cdot}10^{-35},\;\; 1.26{\cdot}10^{-9}\,]^\top$, which is quite a good approximation to $\mathbf{x}^* = \mathbf{0}$. As in Figure 3.3 we see that the ultimate convergence is linear (caused by the singular $\mathbf{J}(\mathbf{x}^*)$), but considerably faster than with the Marquardt method. ∎

Example 3.11. We have used Algorithm 3.21 on the data fitting problem from Examples 1.1, 3.4 and 3.7. As in Example 3.7 we use the starting point $\mathbf{x}_0 \;=\; [-1,\;-2,\;1,\;-1]^\top$, and take $\Delta_0 = 1$ and the stopping criteria given by $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-8}$, $k_{\max} \;=\; 200$. The algorithm stopped after 30 iteration steps with $\mathbf{x} \simeq [-4,\;-5,\;4,\;-4]^\top$. The performance is illustrated below. As in Figure 3.3 we note a very fast ultimate rate of convergence.

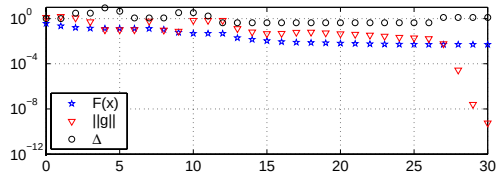

Figure 3.6. *The Dog Leg method applied to the fitting problem from Example 1.1.* ∎

The last two examples seem to indicate that the Dog Leg method is considerably better than the Levenberg-Marquardt method. This is true when the least squares problem arises from a system of nonlinear equations. The Dog Leg method is presently considered as the best method for solving systems of nonlinear equations.

For general least squares problems the Dog Leg method has the same disadvantages as the L-M method: the final convergence can be expected to be linear (and slow) if $F(\mathbf{x}^*) \neq 0$. For a given problem and given starting guess $\mathbf{x}_0$ it is not possible to say beforehand which of the two methods will be the faster.

3.4. A Hybrid Method: L–M and Quasi–Newton

In 1988 Madsen presented a hybrid method which combines the L–M method (quadratic convergence if $F(\mathbf{x}^*) = 0$, linear convergence otherwise) with a Quasi[5]–Newton method, which gives superlinear convergence, even

[5] From Latin: "quasi" = "almost". See Chapter 5 in Frandsen et al (2004) for a general introduction to Quasi–Newton methods.

if $F(\mathbf{x}^*) \neq 0$. The iteration starts with a series of steps with the L-M method. If the performance indicates that $F(\mathbf{x}^*)$ is significantly nonzero, then we switch to the Quasi–Newton method for better performance. It may happen that we get an indication that it is better to switch back to the L–M method, so there is also a mechanism for that.

The switch to the Quasi–Newton method is made if the condition

$$\|\mathbf{F}'(\mathbf{x})\|_\infty < 0.02 * F(\mathbf{x}) \tag{3.22}$$

is satisfied in three consecutive, successful iteration steps. This is interpreted as an indication that we are approaching an $\mathbf{x}^*$ with $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$ and $F(\mathbf{x}^*)$ significantly nonzero. As discussed in connection with (3.12), this can lead to slow, linear convergence.

The Quasi–Newton method is based on having an approximation $\mathbf{B}$ to the Hessian $\mathbf{F}''(\mathbf{x})$ at the current iterate $\mathbf{x}$, and the step $\mathbf{h}_{\text{qn}}$ is found by solving

$$\mathbf{B}\mathbf{h}_{\text{qn}} = -\mathbf{F}'(\mathbf{x}) \; . \tag{3.23}$$

This is an approximation to the Newton equation (2.9a).

The approximation $\mathbf{B}$ is updated by the BFGS strategy, cf Section 5.10 in Frandsen et al (2004): Every $\mathbf{B}$ in the series of approximation matrices is symmetric (as any $\mathbf{F}''(\mathbf{x})$) and positive definite. This ensures that $\mathbf{h}_{\text{qn}}$ is "downhill", cf (2.10). We start with the symmetric, positive definite matrix $\mathbf{B}_0 = \mathbf{I}$, and the BFGS update consists of a rank 2 matrix to be added to the current $\mathbf{B}$. Madsen (1988) uses the following version, advocated by Al-Baali and Fletcher (1985),

$$\begin{aligned} &\mathbf{h} := \mathbf{x}_{\text{new}} - \mathbf{x}; \quad \mathbf{y} := \mathbf{J}_{\text{new}}{}^\top \mathbf{J}_{\text{new}}\mathbf{h} + (\mathbf{J}_{\text{new}} - \mathbf{J})^\top \mathbf{f}(\mathbf{x}_{\text{new}}) \\ &\textbf{if } \mathbf{h}^\top\mathbf{y} > 0 \\ &\quad \mathbf{v} := \mathbf{B}\mathbf{h}; \quad \mathbf{B} := \mathbf{B} + \big(\frac{1}{\mathbf{h}^\top\mathbf{y}}\mathbf{y}\big)\mathbf{y}^\top - \big(\frac{1}{\mathbf{h}^\top\mathbf{v}}\mathbf{v}\big)\mathbf{v}^\top \end{aligned} \tag{3.24}$$

with $\mathbf{J} = \mathbf{J}(\mathbf{x})$, $\mathbf{J}_{\text{new}} = \mathbf{J}(\mathbf{x}_{\text{new}})$. As mentioned, the current $\mathbf{B}$ is positive definite, and it is changed only, if $\mathbf{h}^\top\mathbf{y} > 0$. In this case it can be shown that also the new $\mathbf{B}$ is positive definite.

The Quasi–Newton method is not robust in the global stage of the iteration; it is not guaranteed to be descenting. At the solution $\mathbf{x}^*$ we have $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$, and good final convergence is indicated by rapidly decreasing values of $\|\mathbf{F}'(\mathbf{x})\|$. If these norm values do not decrease rapidly enough, then we switch back to the L–M method.

The algorithm is summarized below. It calls the auxiliary functions *LMstep* and *QNstep*, implementing the two methods.

Algorithm 3.25. A Hybrid Method

begin
  $k := 0$; $\mathbf{x} := \mathbf{x}_0$; $\mu := \mu_0$; $\mathbf{B} := \mathbf{I}$ $\{1^\circ\}$
  found := $(\|\mathbf{F}'(\mathbf{x})\|_\infty \leq \varepsilon_1)$; *method* := L-M
  while (**not** *found*) **and** $(k < k_{\max})$
    $k := k+1$
    case *method* **of**
      L-M:
        $[\mathbf{x}_{\text{new}}, \textit{found}, \textit{better}, \textit{method}, \ldots] := \text{LMstep}(\mathbf{x}, \ldots)$ $\{2^\circ\}$
      Q-N:
        $[\mathbf{x}_{\text{new}}, \textit{found}, \textit{better}, \textit{method}, \ldots] := \text{QNstep}(\mathbf{x}, \mathbf{B}, \ldots)$ $\{2^\circ\}$
    Update $\mathbf{B}$ by (3.24) $\{3^\circ\}$
    if *better*
      $\mathbf{x} := \mathbf{x}_{\text{new}}$
end

We have the following remarks:

1° Initialization. μ_0 can be found by (3.14). The stopping criteria are given by (3.15).

2° The dots indicate that we also transfer current values of $\mathbf{f}$ and $\mathbf{J}$ etc, so that we do not have to recompute them for the same $\mathbf{x}$.

3° Notice that both L-M and Quasi-Newton steps contribute information for the approximation of the Hessian matrix.

The two auxiliary functions are given below,

Function 3.26. Levenberg–Marquardt step

$[\mathbf{x}_{\text{new}}, \mathit{found}, \mathit{better}, \mathit{method}, \ldots] := \text{LMstep}(\mathbf{x}, \ldots)$

begin
  $\mathbf{x}_{\text{new}} := \mathbf{x}$; *method* := L-M
  Solve $(\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}) + \mu\mathbf{I})\mathbf{h}_{\text{lm}} = -\mathbf{F}'(\mathbf{x})$
  if $\|\mathbf{h}_{\text{lm}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$
    found := **true**
  else
    $\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{lm}}$
    $\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\text{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}}))$ $\{4°\}$
    if $\varrho > 0$
      better := **true**; *found* := ($\|\mathbf{F}'(\mathbf{x}_{\text{new}})\|_\infty \le \varepsilon_1$)
      if $\|\mathbf{F}'(\mathbf{x}_{\text{new}})\|_\infty < 0.02 * F(\mathbf{x}_{\text{new}})$ $\{5°\}$
        count := *count*+1
        if *count* = 3 $\{6°\}$
          method := Q-N
      else
        count := 0
    else
      count := 0; *better* := **false**
end

We have the following remarks on the functions *LMstep* and *QNstep*:

4° The gain ratio ϱ is also used to update μ as in Algorithm 3.16.

5° Indication that it might be time to switch method. The parameter *count* is initialized to zero at the start of Algorithm 3.25.

6° (3.22) was satisfied in three consecutive iteration steps, all of which had $\varrho > 0$, ie $\mathbf{x}$ was changed in each of these steps.

Function 3.27. Quasi–Newton step

$[\mathbf{x}_{\text{new}}, \mathit{found}, \mathit{better}, \mathit{method}, \ldots] := \text{QNstep}(\mathbf{x}, \mathbf{B} \ldots)$

begin
  $\mathbf{x}_{\text{new}} := \mathbf{x}$; *method* := Q-N; *better* := **false**
  Solve $\mathbf{B}\mathbf{h}_{\text{qn}} = -\mathbf{F}'(\mathbf{x})$
  if $\|\mathbf{h}_{\text{qn}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$
    found := **true**
  else
    if $\|\mathbf{h}_{\text{qn}}\| > \Delta$ $\{7°\}$
      $\mathbf{h}_{\text{qn}} := (\Delta/\|\mathbf{h}_{\text{qn}}\|) * \mathbf{h}_{\text{qn}}$
    $\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{qn}}$;
    if $\|\mathbf{F}'(\mathbf{x}_{\text{new}})\|_\infty \le \varepsilon_1$ $\{8°\}$
      found := **true**
    else $\{9°\}$
      better := $(F(\mathbf{x}_{\text{new}}) < F(\mathbf{x}))$ **or** $(F(\mathbf{x}_{\text{new}}) \le (1+\delta)F(\mathbf{x})$
                  and $\|\mathbf{F}'(\mathbf{x}_{\text{new}})\|_\infty < \|\mathbf{F}'(\mathbf{x})\|_\infty)$
      if $\|\mathbf{F}'(\mathbf{x}_{\text{new}})\|_\infty \ge \|\mathbf{F}'(\mathbf{x})\|_\infty$ $\{10°\}$
        method := L-M
end

7° We combine the Quasi–Newton method with a trust region approach, with a simple treatment of the case where the bound is active, cf page 15f. At the switch from the L–M method Δ is initialized to $\max\{1.5\varepsilon_2(\|\mathbf{x}\| + \varepsilon_2),\ \frac{1}{5}\|\mathbf{h}_{\text{lm}}\|\}$.

8° Not shown: Δ is updated by means of (2.19).

9° In this part of the algorithm we focus on getting $\mathbf{F}'$ closer to zero, so we accept a slight increase in the value of F, eg $\delta = \sqrt{\varepsilon_{\text{M}}}$, where ε_{M} is the computer's unit roundoff.

10° The gradients do not decrease fast enough.

Example 3.12. Notice that in the updating formula (3.24) the computation of $\mathbf{y}$ involves the product $\mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}_{\text{new}})$. This implies that we have to store the previous Jacobian matrix. Instead, we could use

$$\mathbf{y} \;=\; \mathbf{F}'(\mathbf{x}_{\text{new}}) - \mathbf{F}'(\mathbf{x}) \;=\; \mathbf{g}_{\text{new}} - \mathbf{g}$$

in the updating formula, but Madsen (1988) found that (3.24) performs better.

The trust region approach in the Q–N step was not included in Madsen (1988), but during the development of the immoptibox function `nlshybrid` this idea was found to improve the performance. It reduced the number of times that a Q–N step was tried in vain, ie the condition at 10° immediately returned to the L–M method. ∎

Example 3.13. This hybrid method will not outperform Algorithm 3.16 on the problems discussed in Examples 3.7 and 3.8. In the latter case (see Figure 3.3) $F(\mathbf{x}) \to 0$, and the switching condition at remark 5° will never be satisfied. In the former case, $F(\mathbf{x}^*)$ is significantly nonzero, but – as discussed in Example 3.7 – the simple L–M method has the desired superlinear final convergence.

To demonstrate the efficiency of Algorithm 3.25 we consider the modified *Rosenbrock problem*, cf Example 5.5 in Frandsen et al (1999), given by $\mathbf{f} : \mathbb{R}^2 \mapsto \mathbb{R}^3$,

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

where the parameter λ can be chosen. The minimizer of $F(\mathbf{x}) = \frac{1}{2}\mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$ is $\mathbf{x}^* = [\,1,\ 1\,]^\top$ with $F(\mathbf{x}^*) = \frac{1}{2}\lambda^2$.

Below we give results for Algorithms 3.16 and 3.25 for some values of λ. In all cases we use $\mathbf{x}_0 = [\,-1.2,\ 1\,]^\top$, the initial damping parameter μ_0 defined by $\tau = 10^{-3}$ in (3.14), and $(\varepsilon_1, \varepsilon_2, k_{\max}) = (10^{-10}, 10^{-14}, 200)$ in the stopping criteria (3.15).

In the first two cases λ is too small to really influence the iterations, but for the larger λ-values we see that the hybrid method is much better than the simple Levenberg–Marquardt algorithm – especially as regards the accuracy obtained.

In Figure 3.7 we illustrate the performance of the two algorithms in the case $\lambda = 10^4$.

	Algorithm 3.16		Algorithm 3.25	
λ	its	$\|\mathbf{x} - \mathbf{x}^*\|$	its	$\|\mathbf{x} - \mathbf{x}^*\|$
0	17	2.78e-12	17	2.78e-12
10^{-5}	17	2.78e-12	17	2.78e-12
1	24	1.69e-09	19	2.23e-14
10^2	23	5.87e-07	22	3.16e-12
10^4	23	2.37e-04	22	3.16e-12

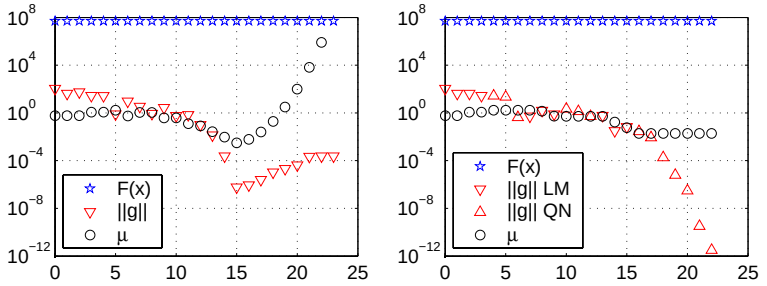

Figure 3.7. *Levenberg–Marquardt's method (left) and the hybrid method (right)*

With the L–M method all steps after no. 15 fail to improve the objective function; μ increases rapidly, and the stopping criterion (3.15b) is satisfied at step no. 23.

With the hybrid method there are several attempts to use the Quasi–Newton method, starting at step nos. 5, 11 and 17. The last attempt is successful, and after 22 steps the iteration is stopped by (3.15a). ∎

3.5. A Secant Version of the L–M Method

The methods discussed in this booklet assume that the vector function $\mathbf{f}$ is differentiable, ie the Jacobian

$$\mathbf{J}(\mathbf{x}) = \left[\frac{\partial f_i}{\partial x_j}\right]$$

exists. In many practical optimization problems it happens that we cannot give formulae for the elements in $\mathbf{J}$, eg because $\mathbf{f}$ is given by a "black box". The secant version of the L–M method is intended for problems of this type.

The simplest remedy is to replace $\mathbf{J}(\mathbf{x})$ by a matrix $\mathbf{B}$ obtained by *numerical differentiation*: The $(i,j)^{\text{th}}$ element is approximated by the finite difference approximation

$$\frac{\partial f_i}{\partial x_j}(\mathbf{x}) \;\simeq\; \frac{f_i(\mathbf{x}+\delta\mathbf{e}_j) - f_i(\mathbf{x})}{\delta} \;\equiv\; b_{ij}\;, \tag{3.28}$$

where $\mathbf{e}_j$ is the unit vector in the jth coordinate direction and δ is an appropriately small real number. With this strategy each iterate $\mathbf{x}$ needs $n+1$ evaluations of $\mathbf{f}$, and since δ is probably much smaller than the distance $\|\mathbf{x}-\mathbf{x}^*\|$, we do not get much more information on the **global** behavior of $\mathbf{f}$ than we would get from just evaluating $\mathbf{f}(\mathbf{x})$. We want better efficiency.

Example 3.14. Let $m=n=1$ and consider one nonlinear equation

$$f:\mathbf{R}\mapsto\mathbf{R}. \qquad \text{Find } \widehat{x} \text{ such that } f(\widehat{x})=0\;.$$

For this problem we can write the Newton–Raphson algorithm (3.6) in the form

$$\begin{aligned} &f(x+h) \;\simeq\; \ell(h) \;\equiv\; f(x)+f'(x)h \\ &\text{solve the linear problem } \; \ell(h)=0 \\ &x_{\text{new}} := x+h \end{aligned} \tag{3.29}$$

If we cannot implement $f'(x)$, then we can approximate it by

$$(f(x+\delta)-f(x))/\delta$$

with δ chosen appropriately small. More generally, we can replace (3.29) by

$$\begin{aligned} &f(x+h) \;\simeq\; \lambda(h) \;\equiv\; f(x)+bh \quad \text{with } \; b\simeq f'(x) \\ &\text{solve the linear problem } \; \lambda(h)=0 \\ &x_{\text{new}} := x+h \end{aligned} \tag{3.30a}$$

Suppose that we already know x_{prev} and $f(x_{\text{prev}})$. Then we can fix the factor b (the approximation to $f'(x)$) by requiring that

$$f(x_{\text{prev}}) = \lambda(x_{\text{prev}}-x)\;. \tag{3.30b}$$

This gives $b = \big(f(x)-f(x_{\text{prev}})\big)/(x-x_{\text{prev}})$, and with this choice of b we recognize (3.30) as the *secant method*, see eg pp 70f in Eldén et al (2004).

The main advantage of the secant method over an alternative finite difference approximation to Newton–Raphson's method is that we only need one function evaluation per iteration step instead of two. ∎

Now, consider the linear model (3.7a) for $\mathbf{f}:\mathbf{R}^n\mapsto\mathbf{R}^m$,

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) \;\simeq\; \boldsymbol{\ell}(\mathbf{h}) \;\equiv\; \mathbf{f}(\mathbf{x})+\mathbf{J}(\mathbf{x})\mathbf{h}\;.$$

We will replace it by

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) \;\simeq\; \boldsymbol{\lambda}(\mathbf{h}) \;\equiv\; \mathbf{f}(\mathbf{x})+\mathbf{B}\mathbf{h}\;,$$

where $\mathbf{B}$ is the current approximation to $\mathbf{J}(\mathbf{x})$. In the next iteration step we need $\mathbf{B}_{\text{new}}$ so that

$$\mathbf{f}(\mathbf{x}_{\text{new}}+\mathbf{h}) \;\simeq\; \mathbf{f}(\mathbf{x}_{\text{new}})+\mathbf{B}_{\text{new}}\mathbf{h}\;.$$

Especially, we want this model to hold with equality for $\mathbf{h}=\mathbf{x}-\mathbf{x}_{\text{new}}$, ie

$$\mathbf{f}(\mathbf{x}) \;=\; \mathbf{f}(\mathbf{x}_{\text{new}})+\mathbf{B}_{\text{new}}(\mathbf{x}-\mathbf{x}_{\text{new}})\;. \tag{3.31a}$$

This gives us m equations in the $m{\cdot}n$ unknown elements of $\mathbf{B}_{\text{new}}$, so we need more conditions. Broyden (1965) suggested to supplement (3.31a) with

$$\mathbf{B}_{\text{new}}\mathbf{v} \;=\; \mathbf{B}\mathbf{v} \quad \text{for all } \; \mathbf{v}\perp(\mathbf{x}-\mathbf{x}_{\text{new}})\;. \tag{3.31b}$$

It is easy to verify that the conditions (3.31a–b) are satisfied by

Definition 3.32. Broyden's Rank One Update

$$\mathbf{B}_{\text{new}} = \mathbf{B}+\mathbf{u}\mathbf{h}^\top$$

where

$$\mathbf{h}=\mathbf{x}_{\text{new}}-\mathbf{x}\;, \quad \mathbf{u}=\frac{1}{\mathbf{h}^\top\mathbf{h}}\left(\mathbf{f}(\mathbf{x}_{\text{new}})-\mathbf{f}(\mathbf{x})-\mathbf{B}\mathbf{h}\right)\;.$$

Note that condition (3.31a) corresponds to the secant condition (3.30b) in the case $n=1$. We say that this approach is a *generalized secant method*.

A brief sketch of the central part of Algorithm 3.16 with this modification has the form

solve $(\mathbf{B}^\top\mathbf{B}+\mu\mathbf{I})\mathbf{h}_{\text{slm}} = -\mathbf{B}^\top\mathbf{f}(\mathbf{x})$
$\mathbf{x}_{\text{new}} := \mathbf{x}+\mathbf{h}_{\text{slm}}$
Update $\mathbf{B}$ by (3.32)
Update μ and $\mathbf{x}$ as in Algorithm 3.16

Powell has shown that if the set of vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ converges to $\mathbf{x}^*$ and if the set of steps $\{\mathbf{h}_k \equiv \mathbf{x}_k - \mathbf{x}_{k-1}\}$ satisfy the condition that $\{\mathbf{h}_{k-n+1}, \ldots, \mathbf{h}_k\}$ are linearly independent (they span the whole of $\mathbb{R}^n$) for each $k \geq n$, then the set of approximations $\{\mathbf{B}_k\}$ converges to $\mathbf{J}(\mathbf{x}^*)$, irrespective of the choice of $\mathbf{B}_0$.

In practice, however, it often happens that the previous n steps do **not** span the whole of $\mathbb{R}^n$, and there is a risk that after some iteration steps the current $\mathbf{B}$ is such a poor approximation to the true Jacobian matrix, that $-\mathbf{B}^\top\mathbf{f}(\mathbf{x})$ is not even a downhill direction. In that case $\mathbf{x}$ will stay unchanged and μ is increased. The approximation $\mathbf{B}$ is changed, but may still be a poor approximation, leading to a further increase in μ, etc. Eventually the process is stopped by $\mathbf{h}_{\text{slm}}$ being so small that (3.15b) is satisfied, although $\mathbf{x}$ may be far from $\mathbf{x}^*$.

A number of strategies have been proposed to overcome this problem, eg to make occasional to recomputations of $\mathbf{B}$ by finite differences. In Algorithm 3.34 below we supplement the updatings determined by the course of the iteration with a cyclic, coordinate-wise series of updatings: Let $\mathbf{h}$ denote the current step, and let j be the current coordinate number. If the angle θ between $\mathbf{h}$ and $\mathbf{e}_j$ is "large", then we compute a finite difference approximation to the j^{th} column of $\mathbf{J}$. More specific, this is done if

$$\cos\theta \;=\; \frac{|\mathbf{h}^\top\mathbf{e}_j|}{\|\mathbf{h}\|\cdot\|\mathbf{e}_j\|} \;<\; \gamma \quad\Leftrightarrow\quad |h_j| \;<\; \gamma\|\mathbf{h}\|\,. \tag{3.33}$$

Experiments indicated that the (rather pessimistic) choice $\gamma = 0.8$ gave good performance. With this choice we can expect that each iteration step needs (almost) two evaluations of the vector function $\mathbf{f}$.

Now we are ready to present the algorithm. The monitoring of the damping parameter μ is as in Algorithm 3.16, and for the sake of clarity we omit it in the presentation.

Algorithm 3.34. Secant Version of the L–M Method

begin
  $k := 0$; $\mathbf{x} := \mathbf{x}_0$; $\mathbf{B} := \mathbf{B}_0$; $j := 0$ $\{1^\circ\}$
  $\mathbf{g} := \mathbf{B}^\top\mathbf{f}(\mathbf{x})$; *found* $= (\ \|\mathbf{g}\|_\infty \leq \varepsilon_1\ )$
  while (**not** *found*) **and** $(k < k_{\max})$
    $k := k+1$; Solve $(\mathbf{B}^\top\mathbf{B}+\mu\mathbf{I})\mathbf{h} = -\mathbf{g}$
    if $\|\mathbf{h}\| \leq \varepsilon_2(\|x\|+\varepsilon_2)$
      found := **true**
    else
      $j := \text{mod}(j,n)+1$; **if** $|h_j| < 0.8\|\mathbf{h}\|$ $\{2^\circ\}$
        Update $\mathbf{B}$ by (3.32), using $\mathbf{x}_{\text{new}} = \mathbf{x}+\eta\mathbf{e}_j$ $\{3^\circ\}$
      $\mathbf{x}_{\text{new}} := \mathbf{x}+\mathbf{h}$; Update $\mathbf{B}$ by (3.32)
      if $F(\mathbf{x}_{\text{new}}) < F(\mathbf{x})$
        $\mathbf{x} := \mathbf{x}_{\text{new}}$
      $\mathbf{g} := \mathbf{B}^\top\mathbf{f}(\mathbf{x})$; *found* $:= (\ \|\mathbf{g}\|_\infty \leq \varepsilon_1\ )$ $\{4^\circ\}$
end

We have the following remarks:

1° Initialization. $\mathbf{x}_0$ is input and $\mathbf{B}_0$ is either input or it is computed by (3.28). Also the parameters in the stopping criteria (3.15) and the step δ to use in (3.28) are input values.

2° Cf (3.33). $\text{mod}(j,n)$ is the remainder after division by n.

3° The step η is given by
  if $x_j = 0$ **then** $\eta := \delta^2$ **else** $\eta := \delta|x_j|$.

4° Whereas the iterate $\mathbf{x}$ is updated only if the descending condition (2.1) is satisfied, the approximation $\mathbf{B}$ is updated in every step. Therefore the approximate gradient $\mathbf{g}$ may change also when $\mathbf{f}(\mathbf{x})$ is unchanged.

Example 3.15. We have used Algorithm 3.34 on the modified Rosenbrock problem from Example 3.13 with $\lambda = 0$. If we use the same starting point and stopping criteria as in that example, and take $\delta = 10^{-7}$ in the difference approximation

(3.28), we find the solution after 29 iteration steps, involving a total of 53 evaluations of $\mathbf{f}(\mathbf{x})$. For comparison, the "true" L–M algorithm needs only 17 steps, implying a total of 18 evaluations of $\mathbf{f}(\mathbf{x})$ **and** $\mathbf{J}(\mathbf{x})$.

We have also used the secant algorithm on the data fitting problem from Examples 1.1, 3.7 and 3.11. With $\delta = 10^{-7}$ and the same starting point and stopping criteria as in Example 3.7 the iteration was stopped by (3.15a) after 94 steps, involving a total of 192 evaluations of $\mathbf{f}(\mathbf{x})$. For comparison, Algorithm 3.16 needs 62 iteration steps.

These two problems indicate that Algorithm 3.34 is robust, but they also illustrate a general rule of thumb: If gradient information is available, it normally pays to use it. ∎

In many applications the numbers m and n are large, but each of the functions $f_i(\mathbf{x})$ depends only on a few of the elements in $\mathbf{x}$. In that case most of the $\frac{\partial f_i}{\partial x_j}(\mathbf{x})$ are zero, and we say that $\mathbf{J}(\mathbf{x})$ is a *sparse matrix*. There are efficient methods exploiting sparsity in the solution of the Levenberg–Marquardt equation (3.13), see eg Nielsen (1997). In the updating formula (3.32), however, normally all elements in the vectors $\mathbf{h}$ and $\mathbf{u}$ are nonzero, so that $\mathbf{B}_{\text{new}}$ will be a *dense matrix*. It is outside the scope of this booklet to discuss how to cope with this; we refer to Gill et al (1984) and Toint (1987).

3.6. A Secant Version of the Dog Leg Method

The idea of using a secant approximation to the Jacobian can, of course, also be used in connection with the Dog Leg Method from Section 3.3. In this section we shall consider the special case of $m = n$, ie in the solution of nonlinear systems of equations. Broyden (1965) not only gave the formula from Definition 3.32,

$$\begin{aligned} &\mathbf{B}_{\text{new}} \;=\; \mathbf{B} + \left(\frac{1}{\mathbf{h}^\top \mathbf{h}} (\mathbf{y} - \mathbf{B}\mathbf{h}) \right) \mathbf{h}^\top \\ &\text{where} \quad \mathbf{h} = \mathbf{x}_{\text{new}} - \mathbf{x} \;, \quad \mathbf{y} = \mathbf{f}(\mathbf{x}_{\text{new}}) - \mathbf{f}(\mathbf{x}) \;, \end{aligned} \tag{3.35a}$$

for updating the approximate Jacobian. He also gave a formula for updating an approximate inverse of the Jacobian, $\mathbf{D} \simeq \mathbf{J}(\mathbf{x})^{-1}$. The formula is

$$\mathbf{D}_{\text{new}} \;=\; \mathbf{D} + \left(\frac{1}{\mathbf{h}^\top \mathbf{D}\mathbf{y}} (\mathbf{h} - \mathbf{D}\mathbf{y}) \right) (\mathbf{h}^\top \mathbf{D}) \;, \tag{3.35b}$$

where $\mathbf{h}$ and $\mathbf{y}$ are defined in (3.35a).

With these matrices the steepest descent direction $\mathbf{h}_{\text{sd}}$ and the Gauss–Newton step (which is identical with the Newton step in this case, cf Example 3.5) $\mathbf{h}_{\text{gn}}$ (3.18) are approximated by

$$\mathbf{h}_{\text{ssd}} = -\mathbf{B}^\top \mathbf{f}(\mathbf{x}) \quad \text{and} \quad \mathbf{h}_{\text{sgn}} = -\mathbf{D}\mathbf{f}(\mathbf{x}) \;. \tag{3.36}$$

Algorithm 3.21 is easily modified to use these approximations. The initial $\mathbf{B} = \mathbf{B}_0$ can be found by the difference approximation (3.28), and $\mathbf{D}_0$ computed as $\mathbf{B}_0^{-1}$. It is easy to show that then the current $\mathbf{B}$ and $\mathbf{D}$ satisfy $\mathbf{B}\mathbf{D} = \mathbf{I}$. The step parameter α is found by (3.19) with $\mathbf{J}(\mathbf{x})$ replaced by $\mathbf{B}$.

Like the secant version of the L–M method, this method needs extra updates to keep $\mathbf{B}$ and $\mathbf{D}$ as good approximations to the current Jacobian and its inverse. We have found that the strategy discussed around (3.33) also works well in this case. It should also be mentioned that the denominator in (3.35b) may be zero or very small. If

$$|\mathbf{h}^\top \mathbf{D}\mathbf{y} < \sqrt{\varepsilon_{\text{M}}}\, \|\mathbf{h}\| \;,$$

then $\mathbf{D}$ is not updated, but computed as $\mathbf{D} = \mathbf{B}^{-1}$.

Each update with (3.35) "costs" $10n^2$ flops[6] and the computation of the two step vectors by (3.36) plus the computation of α by (3.19) costs $6n^2$ flops. Thus, each iteration step with the gradient–free version of the Dog Leg method costs about $16n^2$ flops plus evaluation of $\mathbf{f}(\mathbf{x}_{\text{new}})$. For comparison, each step with Algorithm 3.21 costs about $\frac{2}{3}n^3 + 6n^2$ flops plus evaluation of $\mathbf{f}(\mathbf{x}_{\text{new}})$ **and** $\mathbf{J}(\mathbf{x}_{\text{new}})$. Thus, for large values of n the gradient-free version is cheaper per step. However, the number of iteration steps is often considerably larger, and if the Jacobian matrix is available, then the gradient version is normally faster.

---

[6] One "flop" is a simple arithmetic operation between two floating point numbers.

Example 3.16. We have used Algorithm 3.21 and the gradient–free Dog Leg method on *Rosenbrock's function* $\mathbf{f} : \mathbb{R}^2 \mapsto \mathbb{R}^2$, given by

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

cf Example 3.13. The function has one root, $\mathbf{x}^* = [\, 1,\ 1\,]^\top$, and with both methods we used the starting point $\mathbf{x}_0 = [\, -1.2,\ 1\,]^\top$ and $\varepsilon_1 = \varepsilon_2 = 10^{-12}$, $k_{\max} = 100$ in the stopping criteria (3.15), and $\delta = 10^{-7}$ in (3.28). Algorithm 3.21 stopped at the solution after 17 iteration steps, ie after 18 evaluations of $\mathbf{f}$ and its Jacobian. The secant version also stopped at the solution; this needed 28 iteration steps and a total of 49 evaluations of $\mathbf{f}$. ∎

3.7. Final Remarks

We have discussed a number of algorithms for solving nonlinear least squares problems. All of them appear in any good program library, and implementations can be found via GAMS (Guide to Available Mathematical Software) at the Internet address

http://gams.nist.gov

The examples in this booklet were computed in MATLAB. The programs are available in the toolbox immoptibox, which can be obtained from

http://www.imm.dtu.dk/∼hbn/immoptibox

Finally, it should be mentioned that sometimes a reformulation of the problem can make it easier to solve. We shall illustrate this claim by examples, involving ideas that may be applicable also to **your** problem.

Example 3.17. In Powell's problem from Examples 3.2, 3.8 and 3.10 the variable x_2 occurs only as x_2^2. We can introduce new variables $\mathbf{z} = [\, x_1,\ x_2^2\,]^\top$, and the problem takes the form: Find $\mathbf{z}^* \in \mathbb{R}^2$ such that $\mathbf{f}(\mathbf{z}^*) = \mathbf{0}$, where

$$\mathbf{f}(\mathbf{z}) = \begin{bmatrix} z_1 \\ \frac{10 z_1}{z_1 + 0.1} + 2 z_2 \end{bmatrix} \qquad \text{with} \qquad \mathbf{J}(\mathbf{z}) = \begin{bmatrix} 1 & 0 \\ (z_1 + 0.1)^{-2} & 2 \end{bmatrix} .$$

This Jacobian is nonsingular for all $\mathbf{z}$. The L–M algorithm 3.16 with starting point $\mathbf{z}_0 = [\, 3,\ 1\,]^\top$, $\tau = 10^{-16}$ and $\varepsilon_1 = \varepsilon_2 = 10^{-15}$ in the stopping criteria

(3.15) stops after 3 steps with $\mathbf{z} \simeq$ `[ -1.40e-25, 9.77e-25 ]`$^\top$. This is a good approximation to $\mathbf{z}^* = \mathbf{0}$. ∎

Example 3.18. The data fitting problem from Examples 1.1, 3.7 and 3.11 can be reformulated to have only two parameters, x_1 and x_2: We can write the model in the form

$$M(\mathbf{x}, t) \;=\; c_1 e^{x_1 t} + c_2 e^{x_2 t} ,$$

where, for given $\mathbf{x}$, the vector $\mathbf{c} = \mathbf{c}(\mathbf{x}) \in \mathbb{R}^2$ is found as the least squares solution to the linear problem

$$\mathbf{E}\,\mathbf{c} \;\simeq\; \mathbf{y} ,$$

with $\mathbf{E} = \mathbf{E}(\mathbf{x}) \in \mathbb{R}^{m \times 2}$ given by the rows $(\mathbf{E})_{i,:} = [e^{x_1 t_i} \;\; e^{x_2 t_i}]$. As in Example 1.1 the function $\mathbf{f}$ is defined by $f_i(\mathbf{x}) = y_i - M(\mathbf{x}, t_i)$, leading to

$$\mathbf{f}(\mathbf{x}) = \mathbf{y} - \mathbf{E}(\mathbf{x})\mathbf{c}(\mathbf{x}) .$$

It can be shown that the Jacobian is

$$\mathbf{J} \;=\; -\mathbf{E}\mathbf{G} - \mathbf{H}[\mathbf{c}] ,$$

where, for any vector $\mathbf{u}$ we define the diagonal matrix $[\mathbf{u}] = \mathrm{diag}(\mathbf{u})$, and

$$\mathbf{H} \;=\; [\mathbf{t}]\mathbf{E}, \qquad \mathbf{G} = (\mathbf{E}^\top \mathbf{E})^{-1} \Big([\mathbf{H}^\top \mathbf{f}] - \mathbf{H}^\top \mathbf{E}[\mathbf{c}] \Big) .$$

Algorithm 3.16 with the same poor starting guess as in Example 3.7, $\mathbf{x}_0 = [\, -1,\ -2\,]^\top$, $\tau = 10^{-3}$ and $\varepsilon_1 = \varepsilon_2 = 10^{-8}$ finds the solution $\mathbf{x} \simeq [\, -4,\ -5\,]^\top$ after 13 iteration steps; about $\frac{1}{5}$ of the number of steps needed with the 4-parameter model.

This approach can be generalized to any model, where some of the parameters occur linearly. It has the name *separable least squares*, and is discussed eg in Nielsen (2000) and Golub and Pereyra (2003). ∎

Example 3.19. The final example illustrates a frequent difficulty with least squares problems: Normally the algorithms work best when the problem is scaled so that all the (nonzero) $|x_j^*|$ are of the same order of magnitude.

Consider the so-called *Meyer's problem*

$$f_i(\mathbf{x}) = y_i - x_1 \exp\left(\frac{x_2}{t_i + x_3} \right) , \quad i = 1, \ldots, 16 ,$$

with $t_i = 45+5i$ and

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				

The minimizer is $\mathbf{x}^* \simeq \begin{bmatrix} 5.61{\cdot}10^{-3} & 6.18{\cdot}10^{3} & 3.45{\cdot}10^{2} \end{bmatrix}^\top$ with $F(\mathbf{x}^*) \simeq 43.97$.

An alternative formulation is

$$\phi_i(\mathbf{x}) = 10^{-3} y_i - z_1 \exp\left(\frac{10 z_2}{u_i + z_3} - 13\right) , \quad i = 1, \ldots, 16 ,$$

with $u_i = 0.45+0.05i$. The reformulation corresponds to $\mathbf{z} = \begin{bmatrix} 10^{-3} e^{13} x_1 & 10^{-3} x_2 & 10^{-2} x_3 \end{bmatrix}^\top$, and the minimizer is $\mathbf{z}^* \simeq \begin{bmatrix} 2.48 & 6.18 & 3.45 \end{bmatrix}^\top$ with $\Phi(\mathbf{x}^*) \simeq 4.397{\cdot}10^{-5}$.

If we use Algorithm 3.16 with $\tau = 1$, $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-10}$ and the equivalent starting vectors

$$\mathbf{x}_0 = \begin{bmatrix} 2{\cdot}10^{-2} & 4{\cdot}10^{3} & 2.5{\cdot}10^{2} \end{bmatrix}^\top , \qquad \mathbf{z}_0 = \begin{bmatrix} 8.85 & 4 & 2.5 \end{bmatrix}^\top ,$$

then the iteration is stopped by (3.15b) after 175 iteration steps with the first formulation, and by (3.15a) after 88 steps with the well-scaled reformulation. ∎

APPENDIX

A. Symmetric, Positive Definite Matrices

The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric if $\mathbf{A} = \mathbf{A}^\top$, ie if $a_{ij} = a_{ji}$ for all i, j.

Definition A.1. The symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is

positive definite if $\mathbf{x}^\top \mathbf{A}\, \mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq \mathbf{0}$,

positive semidefinite if $\mathbf{x}^\top \mathbf{A}\, \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq \mathbf{0}$.

Some useful properties of such matrices are listed in Theorem A.2 below. The proof can be found by combining theorems in almost any textbooks on linear algebra and on numerical linear algebra. At the end of this appendix we give some practical implications of the theorem.

Now, let $\mathbf{J} \in \mathbb{R}^{m\times n}$ be given, and let

$$\mathbf{A} = \mathbf{J}^\top \mathbf{J} .$$

Then $\mathbf{A}^\top = \mathbf{J}^\top (\mathbf{J}^\top)^\top = \mathbf{A}$, ie $\mathbf{A}$ is symmetric. Further, for any nonzero $\mathbf{x} \in \mathbb{R}^n$ let $\mathbf{y} = \mathbf{J}\mathbf{x}$. Then

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \mathbf{x}^\top \mathbf{J}^\top \mathbf{J}\, \mathbf{x} = \mathbf{y}^\top \mathbf{y} \geq 0 ,$$

showing that $\mathbf{A}$ is positive semidefinite. If $m \geq n$ and the columns in $\mathbf{J}$ are linearly independent, then $\mathbf{x} \neq \mathbf{0} \ \Rightarrow\ \mathbf{y} \neq \mathbf{0}$ and $\mathbf{y}^\top \mathbf{y} > 0$. Thus, in this case $\mathbf{A}$ is positive definite.

From (A.3) below follows immediately that

$$(\mathbf{A} + \mu \mathbf{I})\mathbf{v}_j = (\lambda_j + \mu)\mathbf{v}_j , \quad j = 1, \ldots, n$$

for any $\mu \in \mathbb{R}$. Combining this with 2° in Theorem A.2 we see that if $\mathbf{A}$ is symmetric and positive semidefinite and $\mu > 0$, then the matrix $\mathbf{A}+\mu\mathbf{I}$ is also symmetric and it is guaranteed to be positive definite.

lower triangular and its diagonal entries are all 1

Theorem A.2. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric and let $\mathbf{A} = \mathbf{LU}$, where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. Further, let $\{(\lambda_j, \mathbf{v}_j)\}_{j=1}^n$ denote the eigensolutions of $\mathbf{A}$, ie

$$\mathbf{A}\mathbf{v}_j \;=\; \lambda_j \mathbf{v}_j \;, \quad j=1,\ldots,n \;. \tag{A.3}$$

Then

1° The eigenvalues are real, $\lambda_j \in \mathbb{R}$, and the eigenvectors $\{\mathbf{v}_j\}$ form an orthonormal basis of $\mathbb{R}^n$.

2° The following statements are equivalent

a) $\mathbf{A}$ is positive definite (positive semidefinite)

b) All $\lambda_j > 0$ ($\lambda_j \geq 0$)

c) All $u_{ii} > 0$ ($u_{ii} \geq 0$) .

If $\mathbf{A}$ is positive definite, then

3° The LU-factorization is numerically stable.

4° $\mathbf{U} = \mathbf{D}\mathbf{L}^\top$ with $\mathbf{D} = \mathrm{diag}(u_{ii})$.

5° $\mathbf{A} = \mathbf{C}^\top\mathbf{C}$, the *Cholesky factorization.* $\mathbf{C} \in \mathbb{R}^{n\times n}$ is upper triangular.

The *condition number* of a symmetric matrix $\mathbf{A}$ is

$$\kappa_2(\mathbf{A}) = \max\{|\lambda_j|\}/\min\{|\lambda_j|\} \;.$$

If $\mathbf{A}$ is positive (semi)definite and $\mu > 0$, then

$$\kappa_2(\mathbf{A}+\mu\mathbf{I}) \;=\; \frac{\max\{\lambda_j\}+\mu}{\min\{\lambda_j\}+\mu} \;\leq\; \frac{\max\{\lambda_j\}+\mu}{\mu} \;,$$

and this is a decreasing function of μ.

Finally, some remarks on Theorem A.2 and practical details: A *unit lower triangular matrix* $\mathbf{L}$ is characterized by $\ell_{ii} = 1$ and $\ell_{ij} = 0$ for $j>i$. Note, that the LU-factorization $\mathbf{A} = \mathbf{LU}$ is made **without** pivoting. Also note that points 4°–5° give the following relation between the LU- and the Cholesky-factorization

$$\mathbf{A} = \mathbf{L}\,\mathbf{U} = \mathbf{L}\,\mathbf{D}\,\mathbf{L}^\top = \mathbf{C}^\top\mathbf{C} \;,$$

showing that

$$\mathbf{C} = \mathbf{D}^{1/2}\mathbf{L}^\top \;, \quad \text{with} \quad \mathbf{D}^{1/2} = \mathrm{diag}(\sqrt{u_{ii}}) \;.$$

The Cholesky factorization can be computed directly (ie without the intermediate results $\mathbf{L}$ and $\mathbf{U}$) by the following algorithm, that includes a test for positive definiteness.

Algorithm A.4. Cholesky Factorization

begin
  $k := 0$; *posdef* := **true** {Initialisation}
  while *posdef* **and** $k < n$
    $k := k+1$; $d := a_{kk} - \sum_{i=1}^{k-1} c_{ik}^2$
    if $d > 0$ {test for pos. def.}
      $c_{kk} := \sqrt{d}$ {diagonal element}
      for $j := k+1,\ldots,n$
        $c_{kj} := \left(a_{kj} - \sum_{i=1}^{k-1} c_{ij}c_{ik}\right)/c_{kk}$ {superdiagonal elements}
    else
      posdef := **false**
end

The "cost" of this algorithm is about $\frac{1}{3}n^3$ flops. Once $\mathbf{C}$ is computed, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be solved by forward and back substitution in

$$\mathbf{C}^\top\mathbf{z} = \mathbf{b} \qquad \text{and} \qquad \mathbf{C}\,\mathbf{x} = \mathbf{z} \;,$$

respectively. Each of these steps costs about n^2 flops.

B. Minimum Norm Least Squares Solution

Consider the least squares problem: Given $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ and $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{h} \in \mathbb{R}^n$ such that

$$\|\mathbf{b} - \mathbf{A}\,\mathbf{h}\|$$

is minimized. To analyze this, we shall use the *singular value decomposition* (SVD) of $\mathbf{A}$,

$$\mathbf{A} \;=\; \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top \;, \tag{B.1}$$

where the matrices $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal, and

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & \mathbf{0} & \end{bmatrix} \quad \text{with } \sigma_1 \geq \cdots \geq \sigma_p > 0, \;\; \sigma_{p+1} = \cdots = \sigma_n = 0 \; .$$

The σ_j are the *singular values*. The number p is the *rank* of $\mathbf{A}$, equal to the dimension of the subspace $\mathcal{R}(\mathbf{A}) \subseteq \mathbb{R}^m$ (the so-called *range* of $\mathbf{A}$) that contains every possible linear combination of the columns of $\mathbf{A}$.

Let $\{\mathbf{u}_j\}_{j=1}^m$ and $\{\mathbf{v}_j\}_{j=1}^n$ denote the columns in $\mathbf{U}$ and $\mathbf{V}$, respectively. Since the matrices are orthogonal, the vectors form two orthonormal sets, ie

$$\mathbf{u}_i^\top \mathbf{u}_j \;=\; \mathbf{v}_i^\top \mathbf{v}_j \;=\; \begin{cases} 1 \; , & i = j \; , \\ 0 \; , & \text{otherwise.} \end{cases} \tag{B.2}$$

From (B.1) and (B.2) it follows that

$$\mathbf{A} = \sum_{j=1}^{p} \sigma_j \mathbf{u}_j \mathbf{v}_j \quad \text{and} \quad \mathbf{A}\,\mathbf{v}_k \;=\; \sigma_k \mathbf{u}_k, \; k = 1, \ldots, n \; . \tag{B.3}$$

The $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ can be used as orthonormal bases in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, so we can write

$$\mathbf{b} \;=\; \sum_{j=1}^{m} \beta_j \mathbf{u}_j, \qquad \mathbf{h} \;=\; \sum_{i=1}^{n} \eta_i \mathbf{v}_i \; , \tag{B.4}$$

and by use of (B.3) we see that

$$\mathbf{r} \;=\; \mathbf{b} - \mathbf{A}\,\mathbf{h} \;=\; \sum_{j=1}^{p} (\beta_j - \sigma_j \eta_j) \mathbf{u}_j + \sum_{j=p+1}^{m} \beta_j \mathbf{u}_j \; ,$$

and by means of (B.2) this implies

$$\|\mathbf{r}\|^2 \;=\; \mathbf{r}^\top \mathbf{r} \;=\; \sum_{j=1}^{p} (\beta_j - \sigma_j \eta_j)^2 + \sum_{j=p+1}^{m} \beta_j^2 \; . \tag{B.5}$$

This is minimized when

$$\beta_j - \sigma_j \eta_j \;=\; 0 \; , \quad j = 1, \ldots, p \; .$$

Thus, the least squares solution can be expressed as

$$\mathbf{h}^* \;= \sum_{j=1}^{p} \frac{\beta_j}{\sigma_j} \mathbf{v}_j + \sum_{j=p+1}^{n} \eta_j \mathbf{v}_j \; .$$

When $p < n$ the least squares solution has $n-p$ degrees of freedom: $\eta_{p+1}, \ldots, \eta_n$ are arbitrary. Similar to the discussion around (B.5) we see that $\|\mathbf{h}^*\|$ is minimized when $\eta_{p+1} = \cdots = \eta_n = 0$. The solution corresponding to this choice of the free parameters is the so-called *minimum norm solution*,

$$\mathbf{h}^*_{\min} \;= \sum_{j=1}^{p} \frac{\beta_j}{\sigma_j} \mathbf{v}_j \;= \sum_{j=1}^{p} \frac{\mathbf{u}_j^\top \mathbf{b}}{\sigma_j} \mathbf{v}_j \; .$$

The reformulation follows from (B.4) and (B.2).

REFERENCES

1. M. Al-Baali and R. Fletcher (1985): *Variational Methods for Non-Linear Least-Squares.* J. Opl. Res. Soc. **36**, No. 5, pp 405–421.
2. C.G. Broyden (1965): *A Class of Methods for Solving Nonlinear Simultaneous Equations.* Maths. Comp. **19**, pp 577–593.
3. J.E. Dennis, Jr. and R.B. Schnabel (1983): *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice Hall.
4. L. Eldén, L. Wittmeyer-Koch and H.B. Nielsen (2004): *Introduction to Numerical Computation – analysis and* MATLAB *illustrations*, to appear.
5. P.E. Frandsen, K. Jonasson, H.B. Nielsen and O. Tingleff (2004): *Unconstrained Optimization, 3rd Edition,* IMM, DTU. Available as http://www.imm.dtu.dk/courses/02611/uncon.pdf
6. P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright (1984): *Sparse Matrix Methods in Optimization,* SIAM J.S.S.C. **5**, pp 562–589.
7. G. Golub and V. Pereyra, (2003): *Separable Nonlinear Least Squares: The Variable Projection Method and its Applications*, Inverse Problems, **19**, pp R1–R26.
8. G. Golub and C.F. Van Loan (1996): *Matrix Computations*, 3rd edition. John Hopkins Press.
9. K. Levenberg (1944): *A Method for the Solution of Certain Problems in Least Squares.* Quart. Appl. Math. **2**, pp 164–168.
10. K. Madsen (1988): *A Combined Gauss-Newton and Quasi-Newton Method for Non-Linear Least Squares.* Institute for Numerical Analysis (now part of IMM), DTU. Report NI-88-10.
11. K. Madsen and H.B. Nielsen (2002): *Supplementary Notes for 02611 Optimization and Data Fitting*, IMM, DTU. Available as http://www.imm.dtu.dk/courses/02611/SN.pdf
12. K. Madsen, H.B. Nielsen and J. Søndergaard (2002): *Robust Subroutines for Non-Linear Optimization.* IMM, DTU. Report IMM-REP-2002-02. Available as http://www.imm.dtu.dk/∼km/F_package/robust.pdf
13. K. Madsen, H.B. Nielsen and O. Tingleff (2004): *Optimization with Constraints, 2nd Edition.* IMM, DTU. Available as http://www.imm.dtu.dk/courses/02611/ConOpt.pdf
14. D. Marquardt (1963): *An Algorithm for Least Squares Estimation on Nonlinear Parameters.* SIAM J. APPL. MATH.**11**, pp 431–441.
15. H.B. Nielsen (1997): *Direct Methods for Sparse Matrices.* IMM, DTU. Available as http://www.imm.dtu.dk/∼hbn/publ/Nsparse.ps
16. H.B. Nielsen (1999): *Damping Parameter in Marquardt's Method.* IMM, DTU. Report IMM-REP-1999-05. Available as http://www.imm.dtu.dk/∼hbn/publ/TR9905.ps
17. H.B. Nielsen (2000): *Separable Nonlinear Least Squares.* IMM, DTU. Report IMM-REP-2000-01. Available as http://www.imm.dtu.dk/∼hbn/publ/TR0001.ps
18. J. Nocedal and S.J. Wright (1999): *Numerical Optimization*, Springer.
19. M.J.D. Powell (1970): *A Hybrid Method for Non-Linear Equations.* In P. Rabinowitz (ed): *Numerical Methods for Non-Linear Algebraic Equations*, Gordon and Breach. pp 87ff.
20. P.L. Toint (1987): *On Large Scale Nonlinear Least Squares Calculations.* SIAM J.S.S.C. **8**, pp 416–435.

INDEX