7

Nonlinear Systems and Numerical Optimization

In this chapter we address the numerical solution of systems of nonlinear equations and the minimization of a function of several variables.

The first problem generalizes to the n-dimensional case the search for the zeros of a function, which was considered in Chapter 6, and can be formulated as follows: given $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$,

$$\text{find } \mathbf{x}^* \in \mathbb{R}^n \text{ such that } \mathbf{F}(\mathbf{x}^*) = \mathbf{0}. \tag{7.1}$$

Problem (7.1) will be solved by extending to several dimensions some of the schemes that have been proposed in Chapter 6.
The basic formulation of the second problem reads: given $f : \mathbb{R}^n \to \mathbb{R}$, called an *objective function*,

$$\text{minimize } f(\mathbf{x}) \quad \text{in } \mathbb{R}^n, \tag{7.2}$$

and is called an *unconstrained optimization problem.*

A typical example consists of determining the optimal allocation of n resources, $x_1, x_2, \ldots, x_n$, in competition with each other and ruled by a specific law. Generally, such resources are not unlimited; this circumstance, from a mathematical standpoint, amounts to requiring that the minimizer of the objective function lies within a subset $\Omega \subset \mathbb{R}^n$, and, possibly, that some equality or inequality constraints must be satisfied.

When these constraints exist the optimization problem is called *constrained* and can be formulated as follows: given the objective function f,

$$\text{minimize } f(\mathbf{x}) \quad \text{in } \Omega \subset \mathbb{R}^n. \tag{7.3}$$

Remarkable instances of (7.3) are those in which Ω is characterized by conditions like $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ (equality constraints) or $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$ (inequality constraints), where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, is a given function, called *cost function*, and the condition $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$ means $h_i(\mathbf{x}) \leq 0$, for $i = 1, \ldots, m$.

If the function $\mathbf{h}$ is continuous and Ω is connected, problem (7.3) is usually referred to as a *nonlinear programming* problem. Notable examples in this area are:

1. *convex programming* if f is a convex function and $\mathbf{h}$ has convex components (see (7.21));
2. *linear programming* if f and $\mathbf{h}$ are linear;
3. *quadratic programming* if f is quadratic and $\mathbf{h}$ is linear.

Problems (7.1) and (7.2) are strictly related to one another. Indeed, if we denote by F_i the components of $\mathbf{F}$, then a point $\mathbf{x}^*$, a solution of (7.1), is a minimizer of the function $f(\mathbf{x}) = \sum_{i=1}^n F_i^2(\mathbf{x})$. Conversely, assuming that f is differentiable and setting the partial derivatives of f equal to zero at a point $\mathbf{x}^*$ at which f is minimum leads to a system of nonlinear equations. Thus, any system of nonlinear equations can be associated with a suitable minimization problem, and vice versa. We shall take advantage of this observation when devising efficient numerical methods.

7.1 Solution of Systems of Nonlinear Equations

Before considering problem (7.1), let us set some notation which will be used throughout the chapter.

For $k \geq 0$, we denote by $C^k(D)$ the set of k-continuously differentiable functions from D to $\mathbb{R}^n$, where $D \subseteq \mathbb{R}^n$ is a set that will be made precise from time to time. We shall always assume that $\mathbf{F} \in C^1(D)$, i.e., $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable function on D.

We denote also by $\mathrm{J}_\mathbf{F}(\mathbf{x})$ the Jacobian matrix associated with $\mathbf{F}$ and evaluated at the point $\mathbf{x} = [x_1, \ldots, x_n]^T$ of $\mathbb{R}^n$, defined as

$$(\mathrm{J}_\mathbf{F}(\mathbf{x}))_{ij} = \left(\frac{\partial F_i}{\partial x_j}\right)(\mathbf{x}), \qquad i, j = 1, \ldots, n.$$

Given any vector norm $\|\cdot\|$, we shall henceforth denote the sphere of radius R with center $\mathbf{x}^*$ by

$$B(\mathbf{x}^*; R) = \{\mathbf{y} \in \mathbb{R}^n : \ \|\mathbf{y} - \mathbf{x}^*\| < R\}.$$

7.1.1 Newton's Method and Its Variants

An immediate extension to the vector case of Newton's method (6.16) for scalar equations can be formulated as follows: given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \ldots$, until convergence

$$\begin{array}{ll} \text{solve} & \mathrm{J}_\mathbf{F}(\mathbf{x}^{(k)})\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)}); \\ \text{set} & \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}. \end{array} \tag{7.4}$$

Thus, at each step k the solution of a linear system with matrix $\mathrm{J}_\mathbf{F}(\mathbf{x}^{(k)})$ is required.

Example 7.1 Consider the nonlinear system

$$\begin{cases} e^{x_1^2+x_2^2} - 1 = 0, \\ e^{x_1^2-x_2^2} - 1 = 0, \end{cases}$$

which admits the unique solution $\mathbf{x}^* = \mathbf{0}$. In this case, $\mathbf{F}(\mathbf{x}) = [e^{x_1^2+x_2^2} - 1, e^{x_1^2-x_2^2} - 1]^T$. Running Program 57, leads to convergence in 15 iterations to the pair $[0.61 \cdot 10^{-5}, 0.61 \cdot 10^{-5}]^T$, starting from the initial datum $\mathbf{x}^{(0)} = [0.1, 0.1]^T$, thus demonstrating a fairly rapid convergence rate. The results, however, dramatically change as the choice of the initial guess is varied. For instance, picking up $\mathbf{x}^{(0)} = [10, 10]^T$, 220 iterations are needed to obtain a solution comparable to the previous one, while, starting from $\mathbf{x}^{(0)} = [20, 20]^T$, Newton's method fails to converge. •

The previous example points out the high sensitivity of Newton's method on the choice of the initial datum $\mathbf{x}^{(0)}$, as confirmed by the following local convergence result.

Theorem 7.1 *Let $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ be a C^1 function in a convex open set D of $\mathbb{R}^n$ that contains $\mathbf{x}^*$. Suppose that $\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)$ exists and that there exist positive constants R, C and L, such that $\|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)\| \leq C$ and*

$$\|\mathrm{J}_{\mathbf{F}}(\mathbf{x}) - \mathrm{J}_{\mathbf{F}}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in B(\mathbf{x}^*; R),$$

having denoted by the same symbol $\|\cdot\|$ two consistent vector and matrix norms. Then, there exists $r > 0$ such that, for any $\mathbf{x}^{(0)} \in B(\mathbf{x}^*; r)$, the sequence (7.4) *is uniquely defined and converges to $\mathbf{x}^*$ with*

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq CL\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2. \tag{7.5}$$

Proof. Proceeding by induction on k, let us check (7.5) and, moreover, that $\mathbf{x}^{(k+1)} \in B(\mathbf{x}^*; r)$, where $r = \min(R, 1/(2CL))$. First, we prove that for any $\mathbf{x}^{(0)} \in B(\mathbf{x}^*; r)$, the inverse matrix $\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})$ exists. Indeed

$$\|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)[\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(0)}) - \mathrm{J}_{\mathbf{F}}(\mathbf{x}^*)]\| \leq \|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)\| \, \|\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(0)}) - \mathrm{J}_{\mathbf{F}}(\mathbf{x}^*)\| \leq CLr \leq \frac{1}{2},$$

and thus, thanks to Theorem 1.5, we can conclude that $\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})$ exists, since

$$\|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})\| \leq \frac{\|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)\|}{1 - \|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)[\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(0)}) - \mathrm{J}_{\mathbf{F}}(\mathbf{x}^*)]\|} \leq 2\|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^*)\| \leq 2C.$$

As a consequence, $\mathbf{x}^{(1)}$ is well defined and

$$\mathbf{x}^{(1)} - \mathbf{x}^* = \mathbf{x}^{(0)} - \mathbf{x}^* - \mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})[\mathbf{F}(\mathbf{x}^{(0)}) - \mathbf{F}(\mathbf{x}^*)].$$

Factoring out $\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})$ on the right-hand side and passing to the norms, we get

$$\begin{aligned} \|\mathbf{x}^{(1)} - \mathbf{x}^*\| &\leq \|\mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(0)})\| \, \|\mathbf{F}(\mathbf{x}^*) - \mathbf{F}(\mathbf{x}^{(0)}) - \mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(0)})[\mathbf{x}^* - \mathbf{x}^{(0)}]\| \\ &\leq 2C\frac{L}{2}\|\mathbf{x}^* - \mathbf{x}^{(0)}\|^2, \end{aligned}$$

where the remainder of Taylor's series of $\mathbf{F}$ has been used. The previous relation proves (7.5) in the case $k = 0$; moreover, since $\mathbf{x}^{(0)} \in B(\mathbf{x}^*; r)$, we have $\|\mathbf{x}^* - \mathbf{x}^{(0)}\| \leq 1/(2CL)$, from which $\|\mathbf{x}^{(1)} - \mathbf{x}^*\| \leq \dfrac{1}{2}\|\mathbf{x}^* - \mathbf{x}^{(0)}\|$.
This ensures that $\mathbf{x}^{(1)} \in B(\mathbf{x}^*; r)$.

By a similar proof, one can check that, should (7.5) be true for a certain k, then the same inequality would follow also for $k+1$ in place of k. This proves the theorem. ◇

Theorem 7.1 thus confirms that Newton's method is quadratically convergent only if $\mathbf{x}^{(0)}$ is sufficiently close to the solution $\mathbf{x}^*$ and if the Jacobian matrix is nonsingular. Moreover, it is worth noting that the computational effort needed to solve the linear system (7.4) can be excessively high as n gets large. Also, $\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(k)})$ could be ill-conditioned, which makes it quite difficult to obtain an accurate solution. For these reasons, several modifications to Newton's method have been proposed, which will be briefly considered in the later sections, referring to the specialized literature for further details (see [OR70], [DS83], [Erh97], [BS90], [SM03], [Deu04] and the references therein).

Remark 7.1 Let $\mathbf{G}(\mathbf{x}) = \mathbf{x} - \mathbf{F}(\mathbf{x})$ and denote by $\mathbf{r}^{(k)} = \mathbf{F}(\mathbf{x}^{(k)})$ the residual at step k. Then, from (7.4) it turns out that Newton's method can be alternatively formulated as

$$\left(\mathrm{I} - \mathrm{J}_{\mathbf{G}}(\mathbf{x}^{(k)})\right)\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = -\mathbf{r}^{(k)},$$

where $\mathrm{J}_{\mathbf{G}}$ denotes the Jacobian matrix associated with $\mathbf{G}$. This equation allows us to interpret Newton's method as a preconditioned stationary Richardson method. This prompts introducing a parameter α_k in order to accelerate the convergence of the iteration

$$\left(\mathrm{I} - \mathrm{J}_{\mathbf{G}}(\mathbf{x}^{(k)})\right)\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = -\alpha_k\mathbf{r}^{(k)}.$$

The problem of how to select α_k will be addressed in Section 7.2.6. ■

7.1.2 Modified Newton's Methods

Several modifications of Newton's method have been proposed in order to reduce its cost when the computed solution is sufficiently close to $\mathbf{x}^*$. Further variants, that are globally convergent, will be introduced for the solution of the minimization problem (7.2).

1. Cyclic updating of the Jacobian matrix

An efficient alternative to method (7.4) consists of keeping the Jacobian matrix (more precisely, its factorization) unchanged for a certain number, say $p \geq 2$, of steps. Generally, a deterioration of convergence rate is accompanied by a gain in computational efficiency.

Program 57 implements Newton's method in the case in which the LU factorization of the Jacobian matrix is updated once every `p` steps. The programs used to solve the triangular systems have been described in Chapter 3.

Here and in later codings in this chapter, we denote by `x0` the initial vector, by `F` and `J` the variables containing the functional expressions of $\mathbf{F}$ and of its Jacobian matrix $\mathrm{J}_{\mathbf{F}}$, respectively. The parameters `tol` and `nmax` represent the stopping tolerance in the convergence of the iterative process and the maximum admissible number of iterations, respectively. In output, the vector `x` contains the approximation to the searched zero of $\mathbf{F}$, while `nit` denotes the number of iterations necessary to converge.

Program 57 - newtonsys : Newton's method for nonlinear systems

```
function [x,iter]=newtonsys(F,J,x0,tol,nmax,p)
%NEWTONSYS Newton method for nonlinear systems
%  [X, ITER] = NEWTONSYS(F, J, X0, TOL, NMAX, P) attempts to solve the
%  nonlinear system F(X)=0 with the Newton method. F and J are strings
%  containing the functional expressions of the nonlinear equations and of
%  the Jacobian matrix. X0 specifies the initial guess. TOL specifies the
%  tolerance of the method. NMAX specifies the maximum number of iterations.
%  P specifies the number of consecutive steps during which the Jacobian is
%  mantained fixed. ITER is the iteration number at which X is computed.
[n,m]=size(F);
if n ~= m, error('Only square systems'); end
iter=0; Fxn=zeros(n,1); x=x0; err=tol+1;
for i=1:n
   for j=1:n
      Jxn(i,j)=eval(J((i-1)*n+j,:));
   end
end
[L,U,P]=lu(Jxn);
step=0;
while err>tol
   if step == p
      step = 0;
      for i=1:n
         Fxn(i)=eval(F(i,:));
         for j=1:n; Jxn(i,j)=eval(J((i-1)*n+j,:)); end
      end
      [L,U,P]=lu(Jxn);
   else
      for i=1:n, Fxn(i)=eval(F(i,:)); end
   end
   iter=iter+1; step=step+1; Fxn=-P*Fxn;
   y=forwardcol(L,Fxn);
   deltax=backwardcol(U,y);
   x = x + deltax;
```

```
        err=norm(deltax);
    if iter > nmax
        error(' Fails to converge within maximum number of iterations ');
    end
end
return
```

2. Inexact solution of the linear systems

Another possibility consists of solving the linear system (7.4) by an iterative method, where the maximum number of admissible iterations is fixed a priori. The resulting schemes are identified as Newton-Jacobi, Newton-SOR or Newton-Krylov methods, according to the iterative process that is used for the linear system (see [BS90], [Kel99]). Here, we limit ourselves to describing the Newton-SOR method.

In analogy with what was done in Section 4.2.1, let us decompose the Jacobian matrix at step k as

$$\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(k)}) = \mathrm{D}_k - \mathrm{E}_k - \mathrm{F}_k, \tag{7.6}$$

where $\mathrm{D}_k = \mathrm{D}(\mathbf{x}^{(k)})$, $-\mathrm{E}_k = -\mathrm{E}(\mathbf{x}^{(k)})$ and $-\mathrm{F}_k = -\mathrm{F}(\mathbf{x}^{(k)})$, the diagonal part and the lower and upper triangular portions of the matrix $\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(k)})$, respectively. We suppose also that D_k is nonsingular. The SOR method for solving the linear system in (7.4) is organized as follows: setting $\boldsymbol{\delta}\mathbf{x}_0^{(k)} = \mathbf{0}$, solve

$$\boldsymbol{\delta}\mathbf{x}_r^{(k)} = \mathrm{M}_k\boldsymbol{\delta}\mathbf{x}_{r-1}^{(k)} - \omega_k(\mathrm{D}_k - \omega_k\mathrm{E}_k)^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \quad r = 1, 2, \ldots, \tag{7.7}$$

where M_k is the iteration matrix of SOR method

$$\mathrm{M}_k = \left[\mathrm{D}_k - \omega_k\mathrm{E}_k\right]^{-1} \ \left[(1-\omega_k)\mathrm{D}_k + \omega_k\mathrm{F}_k\right],$$

and ω_k is a positive relaxation parameter whose optimal value can rarely be determined a priori. Assume that only $r = m$ steps of the method are carried out. Recalling that $\boldsymbol{\delta}\mathbf{x}_r^{(k)} = \mathbf{x}_r^{(k)} - \mathbf{x}^{(k)}$ and still denoting by $\mathbf{x}^{(k+1)}$ the approximate solution computed after m steps, we find that this latter can be written as (see Exercise 1)

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \omega_k\left(\mathrm{M}_k^{m-1} + \ldots + \mathrm{I}\right)\left(\mathrm{D}_k - \omega_k\mathrm{E}_k\right)^{-1}\mathbf{F}(\mathbf{x}^{(k)}). \tag{7.8}$$

This method is thus a composite iteration, in which at each step k, starting from $\mathbf{x}^{(k)}$, m steps of the SOR method are carried out to solve approximately system (7.4).

The integer m, as well as ω_k, can depend on the iteration index k; the simplest choice amounts to performing, at each Newton's step, only one iteration of the SOR method, thus obtaining for $r = 1$ from (7.7) the one-step Newton-SOR method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \omega_k \left(\mathrm{D}_k - \omega_k \mathrm{E}_k\right)^{-1} \mathbf{F}(\mathbf{x}^{(k)}).$$

In a similar way, the preconditioned Newton-Richardson method with matrix P_k, if truncated at the m-th iteration, is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[\mathrm{I} + \mathrm{M}_k + \ldots + \mathrm{M}_k^{m-1}\right] \; \mathrm{P}_k^{-1}\mathbf{F}(\mathbf{x}^{(k)}),$$

where P_k is the preconditioner of $\mathrm{J}_\mathbf{F}$ and

$$\mathrm{M}_k = \mathrm{P}_k^{-1}\mathrm{N}_k, \, \mathrm{N}_k = \mathrm{P}_k - \mathrm{J}_\mathbf{F}(\mathbf{x}^{(k)}).$$

For an efficient implementation of these techniques we refer to the MATLAB software package developed in [Kel99].

3. Difference approximations of the Jacobian matrix

Another possibility consists of replacing $\mathrm{J}_\mathbf{F}(\mathbf{x}^{(k)})$ (whose explicit computation is often very expensive) with an approximation through n-dimensional differences of the form

$$(\mathrm{J}_h^{(k)})_j = \frac{\mathbf{F}(\mathbf{x}^{(k)} + h_j^{(k)}\mathbf{e}_j) - \mathbf{F}(\mathbf{x}^{(k)})}{h_j^{(k)}}, \qquad \forall k \geq 0, \tag{7.9}$$

where $\mathbf{e}_j$ is the j-th vector of the canonical basis of $\mathbb{R}^n$ and $h_j^{(k)} > 0$ are increments to be suitably chosen at each step k of the iteration (7.4). The following result can be shown.

Property 7.1 *Let $\mathbf{F}$ and $\mathbf{x}^*$ be such that the hypotheses of Theorem* 7.1 *are fulfilled, where $\|\cdot\|$ denotes the $\|\cdot\|_1$ vector norm and the corresponding induced matrix norm. If there exist two positive constants ε and h such that $\mathbf{x}^{(0)} \in B(\mathbf{x}^*, \varepsilon)$ and $0 < |h_j^{(k)}| \leq h$ for $j = 1, \ldots, n$ then the sequence defined by*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[\mathrm{J}_h^{(k)}\right]^{-1} \mathbf{F}(\mathbf{x}^{(k)}), \tag{7.10}$$

is well defined and converges linearly to $\mathbf{x}^*$. Moreover, if there exists a positive constant C such that $\max_j |h_j^{(k)}| \leq C\|\mathbf{x}^{(k)} - \mathbf{x}^*\|$ or, equivalently, there exists a positive constant c such that $\max_j |h_j^{(k)}| \leq c\|\mathbf{F}(\mathbf{x}^{(k)})\|$, then the sequence (7.10) *is convergent quadratically.*

This result does not provide any constructive indication as to how to compute the increments $h_j^{(k)}$. In this regard, the following remarks can be made. The first-order truncation error with respect to $h_j^{(k)}$, which arises from the divided difference (7.9), can be reduced by reducing the sizes of $h_j^{(k)}$. On the other hand, a too small value for $h_j^{(k)}$ can lead to large rounding errors. A trade-off must therefore be made between the need of limiting the truncation errors and ensuring a certain accuracy in the computations.

A possible choice is to take

$$h_j^{(k)} = \sqrt{\epsilon_M} \max\left\{|x_j^{(k)}|, M_j\right\} \operatorname{sign}(x_j),$$

where M_j is a parameter that characterizes the typical size of the component x_j of the solution. Further improvements can be achieved using higher-order divided differences to approximate derivatives, like

$$(\mathrm{J}_h^{(k)})_j = \frac{\mathbf{F}(\mathbf{x}^{(k)} + h_j^{(k)}\mathbf{e}_j) - \mathbf{F}(\mathbf{x}^{(k)} - h_j^{(k)}\mathbf{e}_j)}{2h_j^{(k)}}, \qquad \forall k \geq 0.$$

For further details on this subject, see, for instance, [BS90].

7.1.3 Quasi-Newton Methods

By this term, we denote all those schemes in which globally convergent methods are coupled with Newton-like methods that are only locally convergent, but with an order greater than one.
In a quasi-Newton method, given a continuously differentiable function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$, and an initial value $\mathbf{x}^{(0)} \in \mathbb{R}^n$, at each step k one has to accomplish the following operations:

1. compute $\mathbf{F}(\mathbf{x}^{(k)})$;
2. choose $\tilde{\mathrm{J}}_{\mathbf{F}}(\mathbf{x}^{(k)})$ as being either the exact $\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(k)})$ or an approximation of it;
3. solve the linear system $\tilde{\mathrm{J}}_{\mathbf{F}}(\mathbf{x}^{(k)})\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})$;
4. set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \boldsymbol{\delta}\mathbf{x}^{(k)}$, where α_k are suitable *damping parameters.*

Step 4. is thus the characterizing element of this family of methods. It will be addressed in Section 7.2.6, where a criterion for selecting the “direction” $\boldsymbol{\delta}\mathbf{x}^{(k)}$ will be provided.

7.1.4 Secant-like Methods

These methods are constructed starting from the secant method introduced in Section 6.2 for scalar functions. Precisely, given two vectors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$, at the generic step $k \geq 1$ we solve the linear system

$$\mathrm{Q}_k \boldsymbol{\delta}\mathbf{x}^{(k+1)} = -\mathbf{F}(\mathbf{x}^{(k)}) \tag{7.11}$$

and we set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k+1)}$. Q_k is an $n \times n$ matrix such that

$$\mathrm{Q}_k \boldsymbol{\delta}\mathbf{x}^{(k)} = \mathbf{F}(\mathbf{x}^{(k)}) - \mathbf{F}(\mathbf{x}^{(k-1)}) = \mathbf{b}^{(k)}, \qquad k \geq 1,$$

and is obtained by a formal generalization of (6.13). However, the algebraic relation above does not suffice to uniquely determine Q_k. For this purpose we require Q_k for $k \geq n$ to be a solution to the following set of n systems

$$\mathrm{Q}_k\left(\mathbf{x}^{(k)}-\mathbf{x}^{(k-j)}\right)=\mathbf{F}(\mathbf{x}^{(k)})-\mathbf{F}(\mathbf{x}^{(k-j)}), \qquad j=1,\ldots,n. \tag{7.12}$$

If the vectors $\mathbf{x}^{(k-j)}$, ..., $\mathbf{x}^{(k)}$ are linearly independent, system (7.12) allows for calculating all the unknown coefficients $\{(\mathrm{Q}_k)_{lm},\, l,m=1,\ldots,n\}$ of Q_k. Unfortunately, in practice the above vectors tend to become linearly dependent and the resulting scheme is unstable, not to mention the need for storing all the previous n iterates.

For these reasons, an alternative approach is pursued which aims at preserving the information already provided by the method at step k. Precisely, Q_k is looked for in such a way that the difference between the following linear approximants to $\mathbf{F}(\mathbf{x}^{(k-1)})$ and $\mathbf{F}(\mathbf{x}^{(k)})$, respectively

$$\mathbf{F}(\mathbf{x}^{(k)})+\mathrm{Q}_k(\mathbf{x}-\mathbf{x}^{(k)}),\ \mathbf{F}(\mathbf{x}^{(k-1)})+\mathrm{Q}_{k-1}(\mathbf{x}-\mathbf{x}^{(k-1)}),$$

is minimized jointly with the constraint that Q_k satisfies system (7.12). Using (7.12) with $j=1$, the difference between the two approximants is found to be

$$\mathbf{d}_k=(\mathrm{Q}_k-\mathrm{Q}_{k-1})\left(\mathbf{x}-\mathbf{x}^{(k-1)}\right). \tag{7.13}$$

Let us decompose the vector $\mathbf{x}-\mathbf{x}^{(k-1)}$ as

$$\mathbf{x}-\mathbf{x}^{(k-1)}=\alpha\boldsymbol{\delta}\mathbf{x}^{(k)}+\mathbf{s},$$

where $\alpha \in \mathbb{R}$ and $\mathbf{s}^T\boldsymbol{\delta}\mathbf{x}^{(k)}=0$. Therefore, (7.13) becomes

$$\mathbf{d}_k=\alpha\,(\mathrm{Q}_k-\mathrm{Q}_{k-1})\,\boldsymbol{\delta}\mathbf{x}^{(k)}+(\mathrm{Q}_k-\mathrm{Q}_{k-1})\,\mathbf{s}.$$

Only the second term in the relation above can be minimized since the first one is independent of Q_k, being

$$(\mathrm{Q}_k-\mathrm{Q}_{k-1})\boldsymbol{\delta}\mathbf{x}^{(k)}=\mathbf{b}^{(k)}-\mathrm{Q}_{k-1}\boldsymbol{\delta}\mathbf{x}^{(k)}.$$

The problem has thus become: find the matrix Q_k such that $(\mathrm{Q}_k-\mathrm{Q}_{k-1})\,\mathbf{s}$ is minimized $\forall\mathbf{s}$ orthogonal to $\boldsymbol{\delta}\mathbf{x}^{(k)}$ with the constraint that (7.12) holds. It can be shown that such a matrix exists and can be recursively computed as follows

$$\mathrm{Q}_k=\mathrm{Q}_{k-1}+\frac{(\mathbf{b}^{(k)}-\mathrm{Q}_{k-1}\boldsymbol{\delta}\mathbf{x}^{(k)})\boldsymbol{\delta}\mathbf{x}^{(k)^T}}{\boldsymbol{\delta}\mathbf{x}^{(k)^T}\boldsymbol{\delta}\mathbf{x}^{(k)}}. \tag{7.14}$$

The method (7.11), with the choice (7.14) of matrix Q_k is known as the *Broyden method.* To initialize (7.14), we set Q_0 equal to the matrix $\mathrm{J}_{\mathbf{F}}(\mathbf{x}^{(0)})$ or to any approximation of it, for instance, the one yielded by (7.9). As for the convergence of Broyden's method, the following result holds.

Property 7.2 *If the assumptions of Theorem* 7.1 *are satisfied and there exist two positive constants* ε *and* γ *such that*

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\| \leq \varepsilon, \ \|\mathrm{Q}_0 - \mathrm{J}_{\mathbf{F}}(\mathbf{x}^*)\| \leq \gamma,$$

then the sequence of vectors $\mathbf{x}^{(k)}$ *generated by Broyden's method is well defined and converges superlinearly to* $\mathbf{x}^*$*, that is*

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq c_k \|\mathbf{x}^{(k-1)} - \mathbf{x}^*\| \tag{7.15}$$

where the constants c_k *are such that* $\lim\limits_{k\to\infty} c_k = 0$.

Under further assumptions, it is also possible to prove that the sequence Q_k converges to $\mathrm{J}_{\mathbf{F}}(\mathbf{x}^*)$, a property that does not necessarily hold for the above method as demonstrated in Example 7.3.

There exist several variants to Broyden's method which aim at reducing its computational cost, but are usually less stable (see [DS83], Chapter 8). Program 58 implements Broyden's method (7.11)-(7.14). We have denoted by `Q` the initial approximation Q_0 in (7.14).

Program 58 - broyden : Broyden's method for nonlinear systems

```
function [x,iter]=broyden(F,Q,x0,tol,nmax)
%BROYDEN Broyden method for nonlinear systems
%  [X, ITER] = BROYDEN(F, Q, X0, TOL, NMAX) attempts to solve the
%  nonlinear system F(X)=0 with the Broyden method. F is a string variable
%  containing the functional expressions of the nonlinear equations. Q is a
%  starting approximation of the Jacobian. X0 specifies the initial guess.
%  TOL specifies the tolerance of the method. NMAX specifies the maximum
%  number of iterations. ITER is the iteration number at which X is computed.
[n,m]=size(F);
if n ~= m, error('Only square systems'); end
iter=0; err=1+tol; fk=zeros(n,1); fk1=fk; x=x0;
for i=1:n
   fk(i)=eval(F(i,:)); end
   while iter < nmax & err > tol
      s=-Q \ fk;
      x=s+x;
      err=norm(s,inf);
      if err > tol
         for i=1:n,  fk1(i)=eval(F(i,:));  end
         Q=Q+1/(s'*s)*fk1*s';
      end
      iter=iter+1;
      fk=fk1;
   end
end
return
```

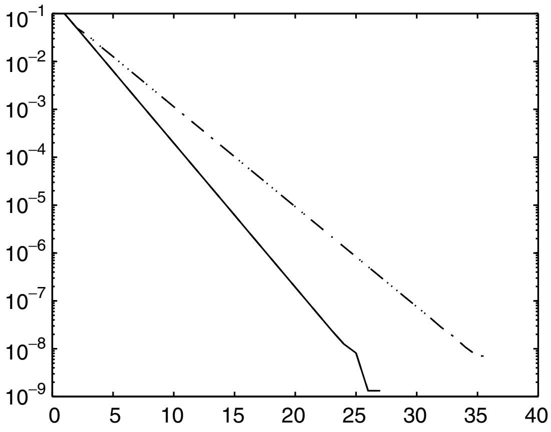

Fig. 7.1. Euclidean norm of the error for the Newton method (*solid line*) and the Broyden method (*dashed line*) in the case of the nonlinear system of Example 7.1

Example 7.2 Let us solve using Broyden's method the nonlinear system of Example 7.1. The method converges in 35 iterations to the value $(0.7 \cdot 10^{-8}, 0.7 \cdot 10^{-8})^T$ compared with the 26 iterations required by Newton's method starting from the same initial guess ($\mathbf{x}^{(0)} = [0.1, 0.1]^T$). The matrix Q_0 has been set equal to the Jacobian matrix evaluated at $\mathbf{x}^{(0)}$. Figure 7.1 shows the behavior of the Euclidean norm of the error for both methods. •

Example 7.3 Suppose we wish to solve using the Broyden method the nonlinear system $\mathbf{F}(\mathbf{x}) = [x_1 + x_2 - 3, x_1^2 + x_2^2 - 9]^T = \mathbf{0}$. This system admits the two solutions $[0, 3]^T$ and $[3, 0]^T$. Broyden's method converges in 8 iterations to the solution $[0, 3]^T$ starting from $\mathbf{x}^{(0)} = [2, 4]^T$. However, the sequence of Q_k, stored in the variable `Q` of Program 58, does not converge to the Jacobian matrix, since

$$\lim_{k\to\infty} \mathrm{Q}^{(k)} = \begin{bmatrix} 1 & 1 \\ 1.5 & 1.75 \end{bmatrix} \neq \mathrm{J}_{\mathbf{F}}([0,3]^T) = \begin{bmatrix} 1 & 1 \\ 0 & 6 \end{bmatrix}.$$

•

7.1.5 Fixed-point Methods

We conclude the analysis of methods for solving systems of nonlinear equations by extending to n-dimensions the fixed-point techniques introduced in the scalar case. For this, we reformulate problem (7.1) as

$$\text{given } \mathbf{G} : \mathbb{R}^n \to \mathbb{R}^n, \text{ find } \mathbf{x}^* \in \mathbb{R}^n \text{ such that } \mathbf{G}(\mathbf{x}^*) = \mathbf{x}^* \tag{7.16}$$

where $\mathbf{G}$ is related to $\mathbf{F}$ through the following property: if $\mathbf{x}^*$ is a fixed point of $\mathbf{G}$, then $\mathbf{F}(\mathbf{x}^*) = \mathbf{0}$.

Analogously to what was done in Section 6.3, we introduce iterative methods for the solution of (7.16) of the form: given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \ldots$ until convergence, find

$$\mathbf{x}^{(k+1)} = \mathbf{G}(\mathbf{x}^{(k)}). \tag{7.17}$$

In order to analyze the convergence of the fixed-point iteration (7.17) the following definition will be useful.

Definition 7.1 A mapping $\mathbf{G} : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is contractive on a set $D_0 \subset D$ if there exists a constant $\alpha < 1$ such that $\|\mathbf{G}(\mathbf{x}) - \mathbf{G}(\mathbf{y})\| \leq \alpha\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}$, $\mathbf{y}$ in D_0, where $\|\cdot\|$ is a suitable vector norm. ■

The existence and uniqueness of a fixed point for $\mathbf{G}$ is ensured by the following theorem.

Theorem 7.2 (Contraction-mapping theorem) *Suppose that* $\mathbf{G} : D \subset \mathbb{R}^n \to \mathbb{R}^n$ *is contractive on a closed set* $D_0 \subset D$ *and that* $\mathbf{G}(\mathbf{x}) \subset D_0$ *for all* $\mathbf{x} \in D_0$. *Then* $\mathbf{G}$ *has a unique fixed point in* D_0.

Proof. Let us first prove the uniqueness of the fixed point. For this, assume that there exist two distinct fixed points, $\mathbf{x}^*$, $\mathbf{y}^*$. Then

$$\|\mathbf{x}^* - \mathbf{y}^*\| = \|\mathbf{G}(\mathbf{x}^*) - \mathbf{G}(\mathbf{y}^*)\| \leq \alpha\|\mathbf{x}^* - \mathbf{y}^*\|$$

from which $(1-\alpha)\|\mathbf{x}^* - \mathbf{y}^*\| \leq 0$. Since $(1-\alpha) > 0$, it must necessarily be that $\|\mathbf{x}^* - \mathbf{y}^*\| = 0$, i.e., $\mathbf{x}^* = \mathbf{y}^*$.

To prove the existence we show that $\mathbf{x}^{(k)}$ given by (7.17) is a Cauchy sequence. This in turn implies that $\mathbf{x}^{(k)}$ is convergent to a point $\mathbf{x}^{(*)} \in D_0$. Take $\mathbf{x}^{(0)}$ arbitrarily in D_0. Then, since the image of $\mathbf{G}$ is included in D_0, the sequence $\mathbf{x}^{(k)}$ is well defined and

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| = \|\mathbf{G}(\mathbf{x}^{(k)}) - \mathbf{G}(\mathbf{x}^{(k-1)})\| \leq \alpha\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|.$$

After p steps, $p \geq 1$, we obtain

$$\begin{aligned}\|\mathbf{x}^{(k+p)} - \mathbf{x}^{(k)}\| &\leq \sum_{i=1}^{p}\|\mathbf{x}^{(k+i)} - \mathbf{x}^{(k+i-1)}\| \leq \left(\alpha^{p-1} + \ldots + 1\right)\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \\ &\leq \frac{\alpha^k}{1-\alpha}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|.\end{aligned}$$

Owing to the continuity of $\mathbf{G}$ it follows that $\lim_{k\to\infty} \mathbf{G}(\mathbf{x}^{(k)}) = \mathbf{G}(\mathbf{x}^{(*)})$ which proves that $\mathbf{x}^{(*)}$ is a fixed point for $\mathbf{G}$. ◇

The following result provides a sufficient condition for the iteration (7.17) to converge (for the proof see [OR70], pp. 299-301), and extends the analogous Theorem 6.3 in the scalar case.

Property 7.3 *Suppose that* $\mathbf{G} : D \subset \mathbb{R}^n \to \mathbb{R}^n$ *has a fixed point* $\mathbf{x}^*$ *in the interior of* D *and that* $\mathbf{G}$ *is continuously differentiable in a neighborhood of* $\mathbf{x}^*$. *Denote by* $\mathrm{J}_{\mathbf{G}}$ *the Jacobian matrix of* $\mathbf{G}$ *and assume that* $\rho(\mathrm{J}_{\mathbf{G}}(\mathbf{x}^{(*)})) < 1$. *Then there exists a neighborhood* S *of* $\mathbf{x}^*$ *such that* $S \subset D$ *and, for any* $\mathbf{x}^{(0)} \in S$, *the iterates defined by* (7.17) *all lie in* D *and converge to* $\mathbf{x}^*$.

As usual, since the spectral radius is the infimum of the induced matrix norms, in order for convergence to hold it suffices to check that $\|\mathrm{J}_{\mathbf{G}}(\mathbf{x})\| < 1$ for some matrix norm.

Example 7.4 Consider the nonlinear system

$$\mathbf{F}(\mathbf{x}) = [x_1^2 + x_2^2 - 1, 2x_1 + x_2 - 1]^T = \mathbf{0},$$

whose solutions are $\mathbf{x}_1^* = [0, 1]^T$ and $\mathbf{x}_2^* = [4/5, -3/5]^T$. To solve it, let us use two fixed-point schemes, respectively defined by the following iteration functions

$$\mathbf{G}_1(\mathbf{x}) = \begin{bmatrix} \dfrac{1 - x_2}{2} \\ \sqrt{1 - x_1^2} \end{bmatrix}, \quad \mathbf{G}_2(\mathbf{x}) = \begin{bmatrix} \dfrac{1 - x_2}{2} \\ -\sqrt{1 - x_1^2} \end{bmatrix}. \tag{7.18}$$

It can be checked that $\mathbf{G}_i(\mathbf{x}_i^*) = \mathbf{x}_i^*$ for $i = 1, 2$ and that the Jacobian matrices of $\mathbf{G}_1$ and $\mathbf{G}_2$, evaluated at $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ respectively, are

$$\mathrm{J}_{\mathbf{G}_1}(\mathbf{x}_1^*) = \begin{bmatrix} 0 & -\frac{1}{2} \\ 0 & 0 \end{bmatrix}, \quad \mathrm{J}_{\mathbf{G}_2}(\mathbf{x}_2^*) = \begin{bmatrix} 0 & -\frac{1}{2} \\ \frac{4}{3} & 0 \end{bmatrix}.$$

The spectral radii are $\rho(\mathrm{J}_{\mathbf{G}_1}(\mathbf{x}_1^*)) = 0$ and $\rho(\mathrm{J}_{\mathbf{G}_2}(\mathbf{x}_2^*)) = \sqrt{2/3} \simeq 0.817 < 1$ so that both methods are convergent in a suitable neighborhood of their respective fixed points.

Running Program 59, with a tolerance of 10^{-10} on the maximum absolute difference between two successive iterates, the first scheme converges to $\mathbf{x}_1^*$ in 9 iterations, starting from $\mathbf{x}^{(0)} = [-0.9, 0.9]^T$, while the second one converges to $\mathbf{x}_2^*$ in 115 iterations, starting from $\mathbf{x}^{(0)} = [0.9, 0.9]^T$. The dramatic change in the convergence behavior of the two methods can be explained in view of the difference between the spectral radii of the corresponding iteration matrices. •

Remark 7.2 Newton's method can be regarded as a fixed-point method with iteration function

$$\mathbf{G}_N(\mathbf{x}) = \mathbf{x} - \mathrm{J}_{\mathbf{F}}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x}). \tag{7.19}$$

■

An implementation of the fixed-point method (7.17) is provided in Program 59. We have denoted by `dim` the size of the nonlinear system and by `Phi` the variables containing the functional expressions of the iteration function $\mathbf{G}$. In output, the vector `alpha` contains the approximation of the sought zero of $\mathbf{F}$ and the vector `res` contains the sequence of the maximum norms of the residuals of $\mathbf{F}(\mathbf{x}^{(k)})$.

Program 59 - fixposys : Fixed-point method for nonlinear systems

```
function [alpha,res,iter]=fixposys(F,Phi,x0,tol,nmax,dim)
%FIXPOSYS Fixed-point method for nonlinear systems
%  [ALPHA, RES, ITER] = FIXPOSYS(F, PHI, X0, TOL, NMAX, DIM) attempts
%  to solve the nonlinear system F(X)=0 with the Fixed Point method. F and PHI are
%  string variables containing the functional expressions of the nonlinear equations
%  and of the iteration function. X0 specifies the initial guess. TOL specifies the
% tolerance of the method. NMAX specifies the maximum number of iterations. DIM is
%  the size of the nonlinear system. ITER is the iteration number at which ALPHA is
%  computed. RES is the system residual computed at ALPHA.
x = x0; alpha=[x']; res = 0;
for k=1:dim
    r=abs(eval(F(k,:))); if (r > res), res = r; end
end;
iter = 0;
residual(1)=res;
while ((iter <= nmax) & (res >= tol)),
   iter = iter + 1;
   for k = 1:dim
       xnew(k) = eval(Phi(k,:));
   end
   x = xnew; res = 0; alpha=[alpha;x]; x=x';
   for k = 1:dim
       r = abs(eval(F(k,:)));
       if (r > res), res=r; end,
   end
   residual(iter+1)=res;
end
res=residual';
return
```

7.2 Unconstrained Optimization

We turn now to minimization problems. The point $\mathbf{x}^*$, the solution of (7.2), is called a *global minimizer* of f, while $\mathbf{x}^*$ is a *local minimizer* of f if $\exists R > 0$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \qquad \forall \mathbf{x} \in B(\mathbf{x}^*; R).$$

Throughout this section we shall always assume that $f \in C^1(\mathbb{R}^n)$, and we refer to [Lem89] for the case in which f is nondifferentiable. We shall denote by

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right)^T,$$

the *gradient* of f at a point $\mathbf{x}$. If $\mathbf{d}$ is a nonnull vector in $\mathbb{R}^n$, then the directional derivative of f with respect to $\mathbf{d}$ is

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \lim_{\alpha \to 0} \frac{f(\mathbf{x} + \alpha \mathbf{d}) - f(\mathbf{x})}{\alpha}$$

and satisfies $\partial f(\mathbf{x})/\partial \mathbf{d} = \nabla f(\mathbf{x})^T \mathbf{d}$. Moreover, denoting by $(\mathbf{x}, \mathbf{x} + \alpha \mathbf{d})$ the segment in $\mathbb{R}^n$ joining the points $\mathbf{x}$ and $\mathbf{x} + \alpha \mathbf{d}$, with $\alpha \in \mathbb{R}$, Taylor's expansion ensures that $\exists \boldsymbol{\xi} \in (\mathbf{x}, \mathbf{x} + \alpha \mathbf{d})$ such that

$$f(\mathbf{x} + \alpha \mathbf{d}) - f(\mathbf{x}) = \alpha \nabla f(\boldsymbol{\xi})^T \mathbf{d}. \tag{7.20}$$

If $f \in C^2(\mathbb{R}^n)$, we shall denote by $\mathrm{H}(\mathbf{x})$ (or $\nabla^2 f(\mathbf{x})$) the *Hessian matrix* of f evaluated at a point $\mathbf{x}$, whose entries are

$$h_{ij}(\mathbf{x}) = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}, \ i, j = 1, \ldots, n.$$

In such a case it can be shown that, if $\mathbf{d} \neq \mathbf{0}$, the second-order directional derivative exists and we have

$$\frac{\partial^2 f}{\partial \mathbf{d}^2}(\mathbf{x}) = \mathbf{d}^T \mathrm{H}(\mathbf{x}) \mathbf{d}.$$

For a suitable $\boldsymbol{\xi} \in (\mathbf{x}, \mathbf{x} + \mathbf{d})$ we also have

$$f(\mathbf{x} + \mathbf{d}) - f(\mathbf{x}) = \nabla f(\mathbf{x})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathrm{H}(\boldsymbol{\xi}) \mathbf{d}.$$

Existence and uniqueness of solutions for (7.2) are not guaranteed in $\mathbb{R}^n$. Nevertheless, the following optimality conditions can be proved.

Property 7.4 *Let $\mathbf{x}^* \in \mathbb{R}^n$ be a local minimizer of f and assume that $f \in C^1(B(\mathbf{x}^*; R))$ for a suitable $R > 0$. Then $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Moreover, if $f \in C^2(B(\mathbf{x}^*; R))$ then $\mathrm{H}(\mathbf{x}^*)$ is positive semidefinite. Conversely, if $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathrm{H}(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is a local minimizer of f in $B(\mathbf{x}^*; R)$.*

A point $\mathbf{x}^*$ such that $\nabla f(\mathbf{x}^*) = \mathbf{0}$, is said to be a *critical point* for f. This condition is necessary for optimality to hold. However, this condition also becomes sufficient if f is a convex function on $\mathbb{R}^n$, i.e., such that $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and for any $\alpha \in [0, 1]$

$$f[\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}] \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}). \tag{7.21}$$

For further and more general existence results, see [Ber82].

7.2.1 Direct Search Methods

In this section we deal with *direct* methods for solving problem (7.2), which only require f to be continuous. In later sections, we shall introduce the so-called *descent* methods, which also involve values of the derivatives of f and have, in general, better convergence properties.

Direct methods are employed when f is not differentiable or if the computation of its derivatives is a nontrivial task. They can also be used to provide an approximate solution to employ as an initial guess for a descent method. For further details, we refer to [Wal75] and [Wol78].

The Hooke and Jeeves Method

Assume we are searching for the minimizer of f starting from a given initial point $\mathbf{x}^{(0)}$ and requiring that the error on the residual is less than a certain fixed tolerance ϵ. The Hooke and Jeeves method computes a new point $\mathbf{x}^{(1)}$ using the values of f at suitable points along the orthogonal coordinate directions around $\mathbf{x}^{(0)}$. The method consists of two steps: an *exploration* step and an *advancing* step.

The exploration step starts by evaluating $f(\mathbf{x}^{(0)} + h_1\mathbf{e}_1)$, where $\mathbf{e}_1$ is the first vector of the canonical basis of $\mathbb{R}^n$ and h_1 is a positive real number to be suitably chosen.

If $f(\mathbf{x}^{(0)} + h_1\mathbf{e}_1) < f(\mathbf{x}^{(0)})$, then a success is recorded and the starting point is moved in $\mathbf{x}^{(0)} + h_1\mathbf{e}_1$, from which an analogous check is carried out at point $\mathbf{x}^{(0)} + h_1\mathbf{e}_1 + h_2\mathbf{e}_2$ with $h_2 \in \mathbb{R}^+$.

If, instead, $f(\mathbf{x}^{(0)} + h_1\mathbf{e}_1) \geq f(\mathbf{x}^{(0)})$, then a failure is recorded and a similar check is performed at $\mathbf{x}^{(0)} - h_1\mathbf{e}_1$. If a success is registered, the method explores, as previously, the behavior of f in the direction $\mathbf{e}_2$ starting from this new point, while, in case of a failure, the method passes directly to examining direction $\mathbf{e}_2$, keeping $\mathbf{x}^{(0)}$ as starting point for the exploration step.

To achieve a certain accuracy, the step lengths h_i must be selected in such a way that the quantities

$$|f(\mathbf{x}^{(0)} \pm h_j\mathbf{e}_j) - f(\mathbf{x}^{(0)}|, \quad j = 1, \ldots, n \tag{7.22}$$

have comparable sizes.

The exploration step terminates as soon as all the n Cartesian directions have been examined. Therefore, the method generates a new point, $\mathbf{y}^{(0)}$, after at most $2n + 1$ functional evaluations. Only two possibilities may arise:

1. $\mathbf{y}^{(0)} = \mathbf{x}^{(0)}$. In such a case, if $\max\limits_{i=1,\ldots,n} h_i \leq \epsilon$ the method terminates and yields the approximate solution $\mathbf{x}^{(0)}$. Otherwise, the step lengths h_i are halved and another exploration step is performed starting from $\mathbf{x}^{(0)}$;
2. $\mathbf{y}^{(0)} \neq \mathbf{x}^{(0)}$. If $\max\limits_{i=1,\ldots,n} |h_i| < \epsilon$, then the method terminates yielding $\mathbf{y}^{(0)}$ as an approximate solution, otherwise the advancing step starts. The advancing step consists of moving further from $\mathbf{y}^{(0)}$ along the direction $\mathbf{y}^{(0)} - \mathbf{x}^{(0)}$

(which is the direction that recorded the maximum decrease of f during the exploration step), rather then simply setting $\mathbf{y}^{(0)}$ as a new starting point $\mathbf{x}^{(1)}$.
This new starting point is instead set equal to $2\mathbf{y}^{(0)}-\mathbf{x}^{(0)}$. From this point a new series of exploration moves is started. If this exploration leads to a point $\mathbf{y}^{(1)}$ such that $f(\mathbf{y}^{(1)}) < f(\mathbf{y}^{(0)} - \mathbf{x}^{(0)})$, then a new starting point for the next exploration step has been found, otherwise the initial guess for further explorations is set equal to $\mathbf{y}^{(1)} = \mathbf{y}^{(0)} - \mathbf{x}^{(0)}$.
The method is now ready to restart from the point $\mathbf{x}^{(1)}$ just computed.

Program 60 provides an implementation of the Hooke and Jeeves method. The input parameters are the size `n` of the problem, the vector `h` of the initial steps along the Cartesian directions, the variable `f` containing the functional expression of f in terms of the components `x(1)`, ..., `x(n)`, the initial point `x0` and the stopping tolerance `tol` equal to ϵ. In output, the code returns the approximate minimizer of f, `x`, the value `minf` attained by f at `x` and the number of iterations needed to compute `x` up to the desired accuracy. The exploration step is performed by Program 61.

Program 60 - hookejeeves : The method of Hooke and Jeeves (HJ)

```
function [x,minf,iter]=hookejeeves(f,n,h,x0,tol)
%HOOKEJEEVES HOOKE and JEEVES method for function minimization.
% [X, MINF, ITER] = HOOKEJEEVES(F, N, H, X0, TOL) attempts to compute the
% minimizer of a function of N variables with the Hooke and Jeeves method. F is
% a string variable containing the functional expression of f. H is an initial
% step. X0 specifies the initial guess. TOL specifies the tolerance of the method.
% ITER is the iteration number at which X is computed. MINF is the value of F at
% the mimimizer X.
x = x0; minf = eval(f); iter = 0;
while h > tol
    [y] = explore(f,n,h,x);
    if y == x
        h = h/2;
    else
        x = 2*y-x;
        [z] = explore(f,n,h,x);
        if z == x
            x = y;
        else
            x = z;
        end
    end
    iter = iter +1;
end
minf = eval(f);
return
```

Program 61 - explore : Exploration step in the HJ method

```
function [x]=explore(f,n,h,x0)
%EXPLORE Exploration step for function minimization.
% [X] = EXPLORE(F, N, H, X0) executes one exploration step of size H in the Hooke
% and Jeeves method for function minimization.
x = x0; f0 = eval(f);
for i=1:n
   x(i) = x(i) + h(i); ff = eval(f);
   if ff < f0
      f0 = ff;
   else
      x(i) = x0(i) - h(i);
      ff = eval(f);
      if ff < f0
         f0 = ff;
      else
         x(i) = x0(i);
      end
   end
end
return
```

The Method of Nelder and Mead

This method, proposed in [NM65], employs local linear approximants of f to generate a sequence of points $\mathbf{x}^{(k)}$, approximations of $\mathbf{x}^*$, starting from simple geometrical considerations. To explain the details of the algorithm, we begin by noticing that a plane in $\mathbb{R}^n$ is uniquely determined by fixing $n+1$ points that must not be lying on a hyperplane.
Denote such points by $\mathbf{x}^{(k)}$, for $k = 0, \ldots, n$. They could be generated as

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + h_k \mathbf{e}_k, \; k = 1, \ldots, n, \tag{7.23}$$

having selected the steplengths $h_k \in \mathbb{R}^+$ in such a way that the variations (7.22) are of comparable size.

Let us now denote by $\mathbf{x}^{(M)}$, $\mathbf{x}^{(m)}$ and $\mathbf{x}^{(\mu)}$ those points of the set $\left\{\mathbf{x}^{(k)}\right\}$ at which f respectively attains its maximum and minimum value and the value immediately preceding the maximum. Moreover, denote by $\mathbf{x}_c^{(k)}$ the *centroid* of point $\mathbf{x}^{(k)}$ defined as

$$\mathbf{x}_c^{(k)} = \frac{1}{n} \sum_{j=0, j\neq k}^{n} \mathbf{x}^{(j)}.$$

The method generates a sequence of approximations of $\mathbf{x}^*$, starting from $\mathbf{x}^{(k)}$, by employing only three possible transformations: *reflections* with respect to centroids, *dilations* and *contractions*. Let us examine the details of the algorithm assuming that $n+1$ initial points are available.

1. Determine the points $\mathbf{x}^{(M)}$, $\mathbf{x}^{(m)}$ and $\mathbf{x}^{(\mu)}$.
2. Compute as an approximation of $\mathbf{x}^*$ the point

$$\bar{\mathbf{x}} = \frac{1}{n+1}\sum_{i=0}^{n}\mathbf{x}^{(i)}$$

and check if $\bar{\mathbf{x}}$ is sufficiently close (in a sense to be made precise) to $\mathbf{x}^*$. Typically, one requires that the standard deviation of the values $f(\mathbf{x}^{(0)}),\ldots,f(\mathbf{x}^{(n)})$ from

$$\bar{f} = \frac{1}{n+1}\sum_{i=0}^{n}f(\mathbf{x}^{(i)})$$

are less than a fixed tolerance ε, that is

$$\frac{1}{n}\sum_{i=0}^{n}\left(f(\mathbf{x}^{(i)})-\bar{f}\right)^2 < \varepsilon.$$

Otherwise, $\mathbf{x}^{(M)}$ is reflected with respect to $\mathbf{x}_c^{(M)}$, that is, the following new point $\mathbf{x}^{(r)}$ is computed

$$\mathbf{x}^{(r)} = (1+\alpha)\mathbf{x}_c^{(M)} - \alpha\mathbf{x}^{(M)},$$

where $\alpha \geq 0$ is a suitable reflection factor. Notice that the method has moved along the "opposite" direction to $\mathbf{x}^{(M)}$. This statement has a geometrical interpretation in the case $n=2$, since the points $\mathbf{x}^{(k)}$ coincide with $\mathbf{x}^{(M)}$, $\mathbf{x}^{(m)}$ and $\mathbf{x}^{(\mu)}$. They thus define a plane whose slope points from $\mathbf{x}^{(M)}$ towards $\mathbf{x}^{(m)}$ and the method provides a step along this direction.
3. If $f(\mathbf{x}^{(m)}) \leq f(\mathbf{x}^{(r)}) \leq f(\mathbf{x}^{(\mu)})$, the point $\mathbf{x}^{(M)}$ is replaced by $\mathbf{x}^{(r)}$ and the algorithm returns to step 2.
4. If $f(\mathbf{x}^{(r)}) < f(\mathbf{x}^{(m)})$ then the reflection step has produced a new minimizer. This means that the minimizer could lie outside the set defined by the convex hull of the considered points. Therefore, this set must be expanded by computing the new vertex

$$\mathbf{x}^{(e)} = \beta\mathbf{x}^{(r)} + (1-\beta)\mathbf{x}_c^{(M)},$$

where $\beta > 1$ is an expansion factor. Then, before coming back to step 2., two possibilities arise:

4a. if $f(\mathbf{x}^{(e)}) < f(\mathbf{x}^{(m)})$ then $\mathbf{x}^{(M)}$ is replaced by $\mathbf{x}^{(e)}$;

4b. $f(\mathbf{x}^{(e)}) \geq f(\mathbf{x}^{(m)})$ then $\mathbf{x}^{(M)}$ is replaced by $\mathbf{x}^{(r)}$ since $f(\mathbf{x}^{(r)}) < f(\mathbf{x}^{(m)})$.

5. If $f(\mathbf{x}^{(r)}) > f(\mathbf{x}^{(\mu)})$ then the minimizer probably lies within a subset of the convex hull of points $\left\{\mathbf{x}^{(k)}\right\}$ and, therefore, two different approaches can be pursued to contract this set. If $f(\mathbf{x}^{(r)}) < f(\mathbf{x}^{(M)})$, the contraction generates a new point of the form

$$\mathbf{x}^{(co)} = \gamma\mathbf{x}^{(r)} + (1-\gamma)\mathbf{x}_c^{(M)}, \quad \gamma \in (0,1),$$

otherwise,

$$\mathbf{x}^{(co)} = \gamma\mathbf{x}^{(M)} + (1-\gamma)\mathbf{x}_c^{(M)}, \quad \gamma \in (0,1),$$

Finally, before returning to step 2., if $f(\mathbf{x}^{(co)}) < f(\mathbf{x}^{(M)})$ and $f(\mathbf{x}^{(co)}) < f(\mathbf{x}^{(r)})$, the point $\mathbf{x}^{(M)}$ is replaced by $\mathbf{x}^{(co)}$, while if $f(\mathbf{x}^{(co)}) \geq f(\mathbf{x}^{(M)})$ or if $f(\mathbf{x}^{(co)}) > f(\mathbf{x}^{(r)})$, then n new points $\mathbf{x}^{(k)}$ are generated, with $k = 1, \ldots, n$, by halving the distances between the original points and $\mathbf{x}^{(0)}$.

As far as the choice of the parameters α, β and γ is concerned, the following values are empirically suggested in [NM65]: $\alpha = 1$, $\beta = 2$ and $\gamma = 1/2$. The resulting scheme is known as the *Simplex method* (that must not be confused with a method sharing the same name used in linear programming), since the set of the points $\mathbf{x}^{(k)}$, together with their convex combinations, form a simplex in $\mathbb{R}^n$.

The convergence rate of the method is strongly affected by the orientation of the starting simplex. To address this concern, in absence of information about the behavior of f, the initial choice (7.23) turns out to be satisfactory in most cases.

We finally mention that the Simplex method is the basic ingredient of the MATLAB function `fmins` for function minimization in n dimensions.

Example 7.5 Let us compare the performances of the Simplex method with the Hooke and Jeeves method, in the minimization of the Rosembrock function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{7.24}$$

This function has a minimizer at $[1,1]^T$ and represents a severe benchmark for testing numerical methods in minimization problems. The starting point for both methods is set equal to $\mathbf{x}^{(0)} = [-1.2, 1]^T$, while the step sizes are taken equal to $h_1 = 0.6$ and $h_2 = 0.5$, in such a way that (7.23) is satisfied. The stopping tolerance on the residual is set equal to 10^{-4}. For the implementation of Simplex method, we have used the MATLAB function `fmins`.

Figure 7.2 shows the iterates computed by the Hooke and Jeeves method (of which one in every ten iterates have been reported, for the sake of clarity) and by the Simplex method, superposed to the level curves of the Rosembrock function. The graph demonstrates the difficulty of this benchmark: actually, the function is like a curved, narrow valley, which attains its minimum along the parabola of equation $x_1^2 - x_2 = 0$.

The Simplex method converges in only 165 iterations, while 935 are needed for the Hooke and Jeeves method to converge. The former scheme yields a solution equal to $[0.999987, 0.999978]^T$, while the latter gives the vector $[0.9655, 0.9322]^T$. •

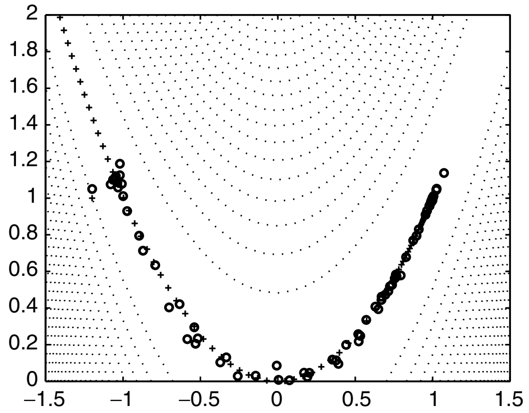

Fig. 7.2. Convergence histories of the Hooke and Jeeves method (*crossed-line*) and the Simplex method (*circled-line*). The level curves of the minimized function (7.24) are reported in dashed line

7.2.2 Descent Methods

In this section we introduce iterative methods that are more sophisticated than those examined in Section 7.2.1. They can be formulated as follows:

given an initial vector $\mathbf{x}^{(0)} \in \mathbb{R}^n$, compute for $k \geq 0$ until convergence

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{7.25}$$

where $\mathbf{d}^{(k)}$ is a suitably chosen direction and α_k is a positive parameter (called *stepsize*) that measures the step along the direction $\mathbf{d}^{(k)}$. This direction $\mathbf{d}^{(k)}$ is a *descent direction* if

$$\begin{aligned} &\mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)}) < 0 \quad && \text{if } \nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}, \\ &\mathbf{d}^{(k)} = \mathbf{0} && \text{if } \nabla f(\mathbf{x}^{(k)}) = \mathbf{0}. \end{aligned} \tag{7.26}$$

A *descent method* is a method like (7.25), in which the vectors $\mathbf{d}^{(k)}$ are descent directions.

Property (7.20) ensures that there exists $\alpha_k > 0$, sufficiently small, such that

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}), \tag{7.27}$$

provided that f is continuously differentiable. Actually, taking in (7.20) $\xi = \mathbf{x}^{(k)} + \vartheta \alpha_k \mathbf{d}^{(k)}$ with $\vartheta \in (0,1)$, and employing the continuity of ∇f, we get

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) - f(\mathbf{x}^{(k)}) = \alpha_k \nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} + \varepsilon, \tag{7.28}$$

where ε tends to zero as α_k tends to zero. As a consequence, if $\alpha_k > 0$ is sufficiently small, the sign of the left-side of (7.28) coincides with the sign of $\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)}$, so that (7.27) is satisfied if $\mathbf{d}^{(k)}$ is a descent direction.

Different choices of $\mathbf{d}^{(k)}$ correspond to different methods. In particular, we recall the following ones:

- *Newton's method*, in which
$$\mathbf{d}^{(k)} = -\mathrm{H}^{-1}(\mathbf{x}^{(k)})\nabla f(\mathbf{x}^{(k)}),$$
provided that H is positive definite within a sufficiently large neighborhood of point $\mathbf{x}^*$;
- *inexact Newton's methods*, in which
$$\mathbf{d}^{(k)} = -\mathrm{B}_k^{-1}\nabla f(\mathbf{x}^{(k)}),$$
where B_k is a suitable approximation of $\mathrm{H}(\mathbf{x}^{(k)})$;
- the *gradient method* or *steepest descent method*, corresponding to setting $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$. This method is thus an inexact Newton's method, in which $\mathrm{B}_k = \mathrm{I}$. It can also be regarded as a gradient-like method, since ${\mathbf{d}^{(k)}}^T\nabla f(\mathbf{x}^{(k)}) = -\|\nabla f(\mathbf{x}^{(k)})\|_2^2$;
- the *conjugate gradient method*, for which
$$\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)}) + \beta_k \mathbf{d}^{(k-1)},$$
where β_k is a scalar to be suitably selected in such a way that the directions $\{\mathbf{d}^{(k)}\}$ turn out to be mutually orthogonal with respect to a suitable scalar product.

Selecting $\mathbf{d}^{(k)}$ is not enough to completely identify a descent method, since it remains an open problem how to determine α_k in such a way that (7.27) is fulfilled without resorting to excessively small stepsizes α_k (and, thus, to methods with a slow convergence).

A method for computing α_k consists of solving the following minimization problem in one dimension:

$$\text{find } \alpha \text{ such that } \phi(\alpha) = f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) \text{ is minimized.} \tag{7.29}$$

In such a case we have the following result.

Theorem 7.3 *Consider the descent method* (7.25)*. If at the generic step k, the parameter α_k is set equal to the exact solution of* (7.29)*, then the following orthogonality property holds*

$$\nabla f(\mathbf{x}^{(k+1)})^T\mathbf{d}^{(k)} = 0.$$

Proof. Let α_k be a solution to (7.29). Then, the first derivative of ϕ, given by

$$\phi'(\alpha) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)})\frac{\partial}{\partial \alpha}(x_i^{(k)} + \alpha d_i^{(k)}) = \nabla f(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)})^T\mathbf{d}^{(k)},$$

vanishes at $\alpha = \alpha_k$. The thesis then follows, recalling the definition of $\mathbf{x}^{(k+1)}$. ◇

Unfortunately, except for in special cases (which are nevetherless quite relevant, see Section 7.2.4), providing an exact solution of (7.29) is not feasible, since this is a nonlinear problem. One possible strategy consists of approximating f along the straight line $\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}$ through an interpolating polynomial and then minimizing this polynomial (see the quadratic interpolation Powell methods and cubic interpolation Davidon methods in [Wal75]).

Generally speaking, a process that leads to an approximate solution to (7.29) is said to be a *line search technique* and is addressed in the next section.

7.2.3 Line Search Techniques

The methods that we are going to deal with in this section, are iterative techniques that terminate as soon as some accuracy stopping criterion on α_k is satisfied. We shall assume that (7.26) holds.

Practical experience reveals that it is not necessary to solve accurately for (7.29) in order to devise efficient methods, rather, it is crucial to enforce some limitation on the step lengths (and, thus, on the admissible values for α_k). Actually, without introducing any limitation, a reasonable request on α_k would seem be that the new iterate $\mathbf{x}^{(k+1)}$ satisfies the inequality

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}), \tag{7.30}$$

where $\mathbf{x}^{(k)}$ and $\mathbf{d}^{(k)}$ have been fixed. For this purpose, the procedure based on starting from a (sufficiently large) value of the step length α_k and halve this value until (7.30) is fulfilled, can yield completely wrong results (see, [DS83]).

More stringent criteria than (7.30) should be adopted in the choice of possible values for α_k. To this end, we notice that two kinds of difficulties arise with the above examples: a slow descent rate of the sequence and the use of small stepsizes.

The first difficulty can be overcome by requiring that

$$\begin{aligned} 0 \geq v_M(\mathbf{x}^{(k+1)}) &= \frac{1}{\alpha_k}\left[f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)})\right] \\ &\geq -\sigma\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}, \end{aligned} \tag{7.31}$$

with $\sigma \in (0, 1/2)$. This amounts to requiring that the average descent rate v_M of f along $\mathbf{d}^{(k)}$, evaluated at $\mathbf{x}^{(k+1)}$, be at least equal to a given fraction of the initial descent rate at $\mathbf{x}^{(k)}$. To avoid the generation of too small stepsizes, we require that the descent rate in the direction $\mathbf{d}^{(k)}$ at $\mathbf{x}^{(k+1)}$ is not less than a given fraction of the descent rate at $\mathbf{x}^{(k)}$

$$|\nabla f(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)})^T\mathbf{d}^{(k)}| \leq \beta|\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}|, \tag{7.32}$$

with $\beta \in (\sigma, 1)$ in such a way as to also satisfy (7.31). In computational practice, $\sigma \in [10^{-5}, 10^{-1}]$ and $\beta \in [10^{-1}, \frac{1}{2}]$ are usual choices. Sometimes, (7.32) is replaced by the milder condition

$$\nabla f(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)})^T\mathbf{d}^{(k)} \geq \beta\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)} \tag{7.33}$$

(recall that $\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}$ is negative, since $\mathbf{d}^{(k)}$ is a descent direction).

The following property ensures that, under suitable assumptions, it is possible to find out values of α_k which satisfy (7.31)-(7.32) or (7.31)-(7.33).

Property 7.5 *Assume that $f(x) \geq M$ for any $x \in \mathbb{R}^n$. Then there exists an interval $I = [c, C]$ for the descent method, with $0 < c < C$, such that $\forall \alpha_k \in I$,* (7.31), (7.32) *(or* (7.31)-(7.33)*) are satisfied, with $\sigma \in (0, 1/2)$ and $\beta \in (\sigma, 1)$.*

Under the constraint of fulfilling conditions (7.31) and (7.32), several choices for α_k are available. Among the most *up-to-date* strategies, we recall here the *backtracking techniques*: having fixed $\sigma \in (0, 1/2)$, then start with $\alpha_k = 1$ and then keep on reducing its value by a suitable scale factor $\rho \in (0, 1)$ (*backtrack* step) until (7.31) is satisfied. This procedure is implemented in Program 62, which requires as input parameters the vector `x` containing $\mathbf{x}^{(k)}$, the macros `f` and `J` of the functional expressions of f and its Jacobian, the vector `d` of the direction $\mathbf{d}^{(k)}$, and a value for σ (usually of the order of 10^{-4}) and the scale factor ρ. In output, the code returns the vector $\mathbf{x}^{(k+1)}$, computed using a suitable value of α_k.

Program 62 - backtrackr : Backtraking for line search

```
function [xnew]= backtrackr(f,J,x,d,sigma,rho)
%BACKTRACKR Backtraking method for line search.
% [XNEW] = BACKTRACKR(F, J, X, D, SIGMA, RHO) attempts to compute the new
% minimizer XNEW with the line search method. F and J are string variables
% containing the functional expressions of f and of its Jacobian. X is the present
% minimizer. D is a given direction. SIGMA and RHO are given parameters.
alphak = 1; fk = eval(f); Jfk = eval (J);
xx = x; x = x + alphak * d; fk1 = eval (f);
while fk1 > fk + sigma * alphak * Jfk'*d
   alphak = alphak*rho;
   x = xx + alphak*d;
   fk1 = eval(f);
end
xnew = x;
return
```

Other commonly used strategies are those developed by *Armijo* and *Goldstein* (see [Arm66], [GP67]). Both use $\sigma \in (0, 1/2)$. In the Armijo formula, one takes $\alpha_k = \beta^{m_k}\bar{\alpha}$, where $\beta \in (0, 1)$, $\bar{\alpha} > 0$ and m_k is the first nonnegative integer such that (7.31) is satisfied. In the Goldstein formula, the parameter α_k is determined in such a way that

$$\sigma \leq \frac{f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) - f(\mathbf{x}^{(k)})}{\alpha_k \nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)}} \leq 1 - \sigma. \tag{7.34}$$

A procedure for computing α_k that satisfies (7.34) is provided in [Ber82], Chapter 1. Of course, one can even choose $\alpha_k = \bar{\alpha}$ for any k, which is clearly convenient when evaluating f is a costly task.

In any case, a good choice of the value $\bar{\alpha}$ is mandatory. In this respect, one can proceed as follows. For a given value $\bar{\alpha}$, the second degree polynomial Π_2 along the direction $\mathbf{d}^{(k)}$ is constructed, subject to the following interpolation constraints

$$\begin{aligned}\Pi_2(\mathbf{x}^{(k)}) &= f(\mathbf{x}^{(k)}),\\ \Pi_2(\mathbf{x}^{(k)}+\bar{\alpha}\mathbf{d}^{(k)}) &= (\mathbf{x}^{(k)}+\bar{\alpha}\mathbf{d}^{(k)}),\\ \Pi_2'(\mathbf{x}^{(k)}) &= \nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}.\end{aligned}$$

Next, the value $\tilde{\alpha}$ is computed such that Π_2 is minimized, then, we let $\bar{\alpha}=\tilde{\alpha}$.

7.2.4 Descent Methods for Quadratic Functions

A case of remarkable interest, where the parameter α_k can be exactly computed, is the problem of minimizing the quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathrm{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{7.35}$$

where A$\in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix and $\mathbf{b}\in\mathbb{R}^n$. In such a case, as already seen in Section 4.3.3, a necessary condition for $\mathbf{x}^*$ to be a minimizer for f is that $\mathbf{x}^*$ is the solution of the linear system (3.2). Actually, it can be checked that if f is a quadratic function

$$\nabla f(\mathbf{x}) = \mathrm{A}\mathbf{x}-\mathbf{b} = -\mathbf{r},\ \mathrm{H}(\mathbf{x}) = \mathrm{A}.$$

As a consequence, all gradient-like iterative methods developed in Section 4.3.3 for linear systems, can be extended *tout-court* to solve minimization problems.

In particular, having fixed a descent direction $\mathbf{d}^{(k)}$, we can determine the optimal value of the acceleration parameter α_k that appears in (7.25), in such a way as to find the point where the function f, restricted to the direction $\mathbf{d}^{(k)}$, is minimized. Setting to zero the directional derivative, we get

$$\frac{\mathrm{d}}{\mathrm{d}\alpha_k} f(\mathbf{x}^{(k)}+\alpha_k\mathbf{d}^{(k)}) = -{\mathbf{d}^{(k)}}^T\mathbf{r}^{(k)} + \alpha_k{\mathbf{d}^{(k)}}^T\mathrm{A}\mathbf{d}^{(k)} = 0$$

from which the following expression for α_k is obtained

$$\alpha_k = \frac{{\mathbf{d}^{(k)}}^T\mathbf{r}^{(k)}}{{\mathbf{d}^{(k)}}^T\mathrm{A}\mathbf{d}^{(k)}}. \tag{7.36}$$

The error introduced by the iterative process (7.25) at the k-th step is

$$\begin{aligned}\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|_\mathrm{A}^2 &= \left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)^T\mathrm{A}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\\ &= \|\mathbf{x}^{(k)}-\mathbf{x}^*\|_\mathrm{A}^2 + 2\alpha_k{\mathbf{d}^{(k)}}^T\mathrm{A}\left(\mathbf{x}^{(k)}-\mathbf{x}^*\right)\\ &\quad +\alpha_k^2{\mathbf{d}^{(k)}}^T\mathrm{A}\mathbf{d}^{(k)}.\end{aligned} \tag{7.37}$$

On the other hand $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathrm{A}}^2 = \mathbf{r}^{(k)^T}\mathrm{A}^{-1}\mathbf{r}^{(k)}$, so that from (7.37) it follows that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|_{\mathrm{A}}^2 = \rho_k \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathrm{A}}^2, \tag{7.38}$$

having denoted by $\rho_k = 1 - \sigma_k$, with

$$\sigma_k = (\mathbf{d}^{(k)^T}\mathbf{r}^{(k)})^2 / \left(\left(\mathbf{d}^{(k)}\right)^T \mathrm{A}\mathbf{d}^{(k)} \left(\mathbf{r}^{(k)}\right)^T \mathrm{A}^{-1}\mathbf{r}^{(k)} \right).$$

Since A is symmetric and positive definite, σ_k is always positive. Moreover, it can be directly checked that ρ_k is strictly less than 1, except when $\mathbf{d}^{(k)}$ is orthogonal to $\mathbf{r}^{(k)}$, in which case $\rho_k = 1$.

The choice $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$, which leads to the *steepest descent method*, prevents this last circumstance from arising. In such a case, from (7.38) we get

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|_{\mathrm{A}} \leq \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathrm{A}} \tag{7.39}$$

having employed the following result.

Lemma 7.1 (Kantorovich inequality) *Let* $\mathrm{A} \in \mathbb{R}^{n\times n}$ *be a symmetric positive definite matrix whose eigenvalues with largest and smallest module are given by* λ_{max} *and* λ_{min}*, respectively. Then,* $\forall \mathbf{y} \in \mathbb{R}^n$*,* $\mathbf{y} \neq \mathbf{0}$*,*

$$\frac{(\mathbf{y}^T\mathbf{y})^2}{(\mathbf{y}^T\mathrm{A}\mathbf{y})(\mathbf{y}^T\mathrm{A}^{-1}\mathbf{y})} \geq \frac{4\lambda_{max}\lambda_{min}}{(\lambda_{max} + \lambda_{min})^2}.$$

It follows from (7.39) that, if A is ill-conditioned, the error reducing factor for the *steepest descent* method is close to 1, yielding a slow convergence to the minimizer $\mathbf{x}^*$. As done in Chapter 4, this drawback can be overcome by introducing directions $\mathbf{d}^{(k)}$ that are mutually A-conjugate, i.e.

$$\mathbf{d}^{(k)^T}\mathrm{A}\mathbf{d}^{(m)} = 0 \text{ if } k \neq m.$$

The corresponding methods enjoy the following finite termination property.

Property 7.6 *A method for computing the minimizer* $\mathbf{x}^*$ *of the quadratic function* (7.35) *which employs* A*-conjugate directions terminates after at most* n *steps if the acceleration parameter* α_k *is selected as in* (7.36)*. Moreover, for any* k*,* $\mathbf{x}^{(k+1)}$ *is the minimizer of* f *over the subspace generated by the vectors* $\mathbf{x}^{(0)}, \mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(k)}$ *and*

$$\mathbf{r}^{(k+1)^T}\mathbf{d}^{(m)} = 0 \; \forall m \leq k.$$

The A-conjugate directions can be determined by following the procedure described in Section 4.3.4. Given $\mathbf{x}^{(0)} \in \mathbb{R}^n$ and letting $\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$, the *conjugate gradient method* for function minimization is

$$\begin{aligned}
\mathbf{d}^{(k+1)} &= \mathbf{r}^{(k)} + \beta_k \mathbf{d}^{(k)}, \\
\beta_k &= -\frac{\mathbf{r}^{(k+1)^T} \mathrm{A} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)^T} \mathrm{A} \mathbf{d}^{(k)}} = \frac{\mathbf{r}^{(k+1)^T} \mathbf{r}^{(k+1)}}{\mathbf{r}^{(k)^T} \mathbf{r}^{(k)}}, \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.
\end{aligned}$$

It satisfies the following error estimate

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathrm{A}} \leq 2 \left(\frac{\sqrt{K_2(\mathrm{A})} - 1}{\sqrt{K_2(\mathrm{A})} + 1} \right)^k \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_{\mathrm{A}},$$

which can be improved by lowering the condition number of A, i.e., resorting to the preconditioning techniques that have been dealt with in Section 4.3.2.

Remark 7.3 (The nonquadratic case) The conjugate gradient method can be extended to the case in which f is a nonquadratic function. However, in such an event, the acceleration parameter α_k cannot be exactly determined a priori, but requires the solution of a local minimization problem. Moreover, the parameters β_k can no longer be uniquely found. Among the most reliable formulae, we recall the one due to Fletcher-Reeves,

$$\beta_1 = 0, \, \beta_k = \frac{\|\nabla f(\mathbf{x}^{(k)})\|_2^2}{\|\nabla f(\mathbf{x}^{(k-1)})\|_2^2}, \quad \text{for } k > 1$$

and the one due to Polak-Ribiére

$$\beta_1 = 0, \, \beta_k = \frac{\nabla f(\mathbf{x}^{(k)})^T (\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)}))}{\|\nabla f(\mathbf{x}^{(k-1)})\|_2^2}, \qquad \text{for } k > 1.$$

■

7.2.5 Newton-like Methods for Function Minimization

Another example of descent method is provided by Newton's method, which differs from its version for nonlinear systems in that now it is no longer applied to f, but to its gradient.

Using the notation of Section 7.2.2, Newton's method for function minimization amounts to computing, given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \ldots,$ until convergence

$$\begin{aligned}
\mathbf{d}^{(k)} &= -\mathrm{H}_k^{-1} \nabla f(\mathbf{x}^{(k)}), \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{d}^{(k)},
\end{aligned} \tag{7.40}$$

having set $\mathrm{H}_k = \mathrm{H}(\mathbf{x}^{(k)})$. The method can be derived by truncating Taylor's expansion of $f(\mathbf{x}^{(k)})$ at the second-order,

$$f(\mathbf{x}^{(k)} + \mathbf{p}) \simeq f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \mathrm{H}_k \mathbf{p}. \qquad (7.41)$$

Selecting $\mathbf{p}$ in (7.41) in such a way that the new vector $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{p}$ satisfies $\nabla f(\mathbf{x}^{k+1}) = \mathbf{0}$, we end up with method (7.40), which thus converges in one step if f is quadratic.

In the general case, a result analogous to Theorem 7.1 also holds for function minimization. Method (7.40) is therefore locally quadratically convergent to the minimizer $\mathbf{x}^*$. However, it is not convenient to use Newton's method from the beginning of the computation, unless $\mathbf{x}^{(0)}$ is sufficiently close to $\mathbf{x}^*$. Otherwise, indeed, H_k could not be invertible and the directions $\mathbf{d}^{(k)}$ could fail to be descent directions. Moreover, if H_k is not positive definite, nothing prevents the scheme (7.40) from converging to a saddle point or a maximizer, which are points where ∇f is equal to zero. All these drawbacks, together with the high computational cost (recall that a linear system with matrix H_k must be solved at each iteration), prompt suitably modifying method (7.40), which leads to the so-called *quasi-Newton* methods.

A first modification, which applies to the case where H_k is not positive definite, yields the so-called *Newton's method with shift.* The idea is to prevent Newton's method from converging to non-minimizers of f, by applying the scheme to a new Hessian matrix $\tilde{\mathrm{H}}_k = \mathrm{H}_k + \mu_k \mathrm{I}_n$, where, as usual, I_n denotes the identity matrix of order n and μ_k is selected in such a way that $\tilde{\mathrm{H}}_k$ is positive definite. The problem is to determine the *shift* μ_k with a reduced effort. This can be done, for instance, by applying the Gershgorin theorem to the matrix $\tilde{\mathrm{H}}_k$ (see Section 5.1). For further details on the subject, see [DS83] and [GMW81].

7.2.6 Quasi-Newton Methods

At the generic k-th iteration, a *quasi-Newton method* for function minimization performs the following steps:

1. compute the Hessian matrix H_k, or a suitable approximation B_k;
2. find a descent direction $\mathbf{d}^{(k)}$ (not necessarily coinciding with the direction provided by Newton's method), using H_k or B_k;
3. compute the acceleration parameter α_k;
4. update the solution, setting $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$, according to a global convergence criterion.

In the particular case where $\mathbf{d}^{(k)} = -\mathrm{H}_k^{-1}\nabla f(\mathbf{x}^{(k)})$, the resulting scheme is called the *damped Newton's method.* To compute H_k or B_k, one can resort to either Newton's method or secant-like methods, which will be considered in Section 7.2.7.

The criteria for selecting the parameter α_k, that have been discussed in Section 7.2.3, can now be usefully employed to devise globally convergent methods. Property 7.5 ensures that there exist values of α_k satisfying (7.31), (7.33) or (7.31), (7.32).

Let us then assume that a sequence of iterates $\mathbf{x}^{(k)}$, generated by a descent method for a given $\mathbf{x}^{(0)}$, converge to a vector $\mathbf{x}^*$. This vector will not be, in general, a critical point for f. The following result gives some conditions on the directions $\mathbf{d}^{(k)}$ which ensure that the limit $\mathbf{x}^*$ of the sequence is also a critical point of f.

Property 7.7 (Convergence) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function, and assume that there exists $L > 0$ such that*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2.$$

Then, if $\left\{\mathbf{x}^{(k)}\right\}$ is a sequence generated by a gradient-like method which fulfills (7.31) *and* (7.33)*, then, one (and only one) of the following events can occur:*

1. $\nabla f(\mathbf{x}^{(k)}) = \mathbf{0}$ *for some k;*
2. $\lim\limits_{k\to\infty} f(\mathbf{x}^{(k)}) = -\infty$*;*
3. $\lim\limits_{k\to\infty} \dfrac{\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)}}{\|\mathbf{d}^{(k)}\|_2} = 0.$

Thus, unless the pathological cases where the directions $\mathbf{d}^{(k)}$ become too large or too small with respect to $\nabla f(\mathbf{x}^{(k)})$ or, even, are orthogonal to $\nabla f(\mathbf{x}^{(k)})$, any limit of the sequence $\left\{\mathbf{x}^{(k)}\right\}$ is a critical point of f.

The convergence result for the sequence $\mathbf{x}^{(k)}$ can also be extended to the sequence $f(\mathbf{x}^{(k)})$. Indeed, the following result holds.

Property 7.8 *Let $\left\{\mathbf{x}^{(k)}\right\}$ be a convergent sequence generated by a gradient-like method, i.e., such that any limit of the sequence is also a critical point of f. If the sequence $\left\{\mathbf{x}^{(k)}\right\}$ is bounded, then $\nabla f(\mathbf{x}^{(k)})$ tends to zero as $k \to \infty$.*

For the proofs of the above results, see [Wol69] and [Wol71].

7.2.7 Secant-like methods

In quasi-Newton methods the Hessian matrix H is replaced by a suitable approximation. Precisely, the generic iterate is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathrm{B}_k^{-1} \nabla f(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}.$$

Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is of class C^2 on an open convex set $D \subset \mathbb{R}^n$. In such a case, H is symmetric and, as a consequence, approximants B_k of H ought to be symmetric. Moreover, if B_k were symmetric at a point $\mathbf{x}^{(k)}$, we would also like the next approximant B_{k+1} to be symmetric at $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$.
To generate B_{k+1} starting from B_k, consider the Taylor expansion

$$\nabla f(\mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k+1)}) + \mathrm{B}_{k+1}(\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}),$$

from which we get

$$\mathrm{B}_{k+1}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}, \text{ with } \mathbf{y}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}).$$

Using again a series expansion of B, we end up with the following first-order approximation of H

$$\mathrm{B}_{k+1} = \mathrm{B}_k + \frac{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})\mathbf{c}^T}{\mathbf{c}^T\mathbf{s}^{(k)}}, \tag{7.42}$$

where $\mathbf{c} \in \mathbb{R}^n$ and having assumed that $\mathbf{c}^T\mathbf{s}^{(k)} \neq 0$. We notice that taking $\mathbf{c} = \mathbf{s}^{(k)}$ yields Broyden's method, already discussed in Section 7.1.4 in the case of systems of nonlinear equations.

Since (7.42) does not guarantee that B_{k+1} is symmetric, it must be suitably modified. A way for constructing a symmetric approximant B_{k+1} consists of choosing $\mathbf{c} = \mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)}$ in (7.42), assuming that $(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})^T\mathbf{s}^{(k)} \neq 0$. By so doing, the following symmetric first-order approximation is obtained

$$\mathrm{B}_{k+1} = \mathrm{B}_k + \frac{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})^T}{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})^T\mathbf{s}^{(k)}}. \tag{7.43}$$

From a computational standpoint, disposing of an approximation for H is not completely satisfactory, since the inverse of the approximation of H appears in the iterative methods that we are dealing with. Using the Sherman-Morrison formula (3.57), with $\mathrm{C}_k = \mathrm{B}_k^{-1}$, yields the following recursive formula for the computation of the inverse

$$\mathrm{C}_{k+1} = \mathrm{C}_k + \frac{(\mathbf{s}^{(k)} - \mathrm{C}_k\mathbf{y}^{(k)})(\mathbf{s}^{(k)} - \mathrm{C}_k\mathbf{y}^{(k)})^T}{(\mathbf{s}^{(k)} - \mathrm{C}_k\mathbf{y}^{(k)})^T\mathbf{y}^{(k)}}, \; k = 0, 1, \ldots, \tag{7.44}$$

having assumed that $\mathbf{y}^{(k)} = \mathrm{B}\mathbf{s}^{(k)}$, where B is a symmetric nonsingular matrix, and that $(\mathbf{s}^{(k)} - \mathrm{C}_k\mathbf{y}^{(k)})^T\mathbf{y}^{(k)} \neq 0$.

An algorithm that employs the approximations (7.43) or (7.44), is potentially unstable when $(\mathbf{s}^{(k)} - \mathrm{C}_k\mathbf{y}^{(k)})^T\mathbf{y}^{(k)} \simeq 0$, due to rounding errors. For this reason, it is convenient to set up the previous scheme in a more stable form. To this end, instead of (7.42), we introduce the approximation

$$\mathrm{B}_{k+1}^{(1)} = \mathrm{B}_k + \frac{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})\mathbf{c}^T}{\mathbf{c}^T\mathbf{s}^{(k)}},$$

then, we define $\mathrm{B}_{k+1}^{(2)}$ as being the symmetric part

$$\mathrm{B}_{k+1}^{(2)} = \frac{\mathrm{B}_{k+1}^{(1)} + (\mathrm{B}_{k+1}^{(1)})^T}{2}.$$

The procedure can be iterated as follows

$$\begin{aligned}
\mathrm{B}_{k+1}^{(2j+1)} &= \mathrm{B}_{k+1}^{(2j)} + \frac{(\mathbf{y}^{(k)} - \mathrm{B}_{k+1}^{(2j)}\mathbf{s}^{(k)})\mathbf{c}^T}{\mathbf{c}^T\mathbf{s}^{(k)}}, \\
\mathrm{B}_{k+1}^{(2j+2)} &= \frac{\mathrm{B}_{k+1}^{(2j+1)} + (\mathrm{B}_{k+1}^{(2j+1)})^T}{2},
\end{aligned} \tag{7.45}$$

with $k = 0, 1, \dots$ and having set $\mathrm{B}_{k+1}^{(0)} = \mathrm{B}_k$. It can be shown that the limit as j tends to infinity of (7.45) is

$$\begin{aligned}
\lim_{j\to\infty} \mathrm{B}_{k+1}^{(j)} = \mathrm{B}_{k+1} = \mathrm{B}_k &+ \frac{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})\mathbf{c}^T + \mathbf{c}(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})^T}{\mathbf{c}^T\mathbf{s}^{(k)}} \\
&- \frac{(\mathbf{y}^{(k)} - \mathrm{B}_k\mathbf{s}^{(k)})^T\mathbf{s}^{(k)}}{(\mathbf{c}^T\mathbf{s}^{(k)})^2}\mathbf{c}\mathbf{c}^T,
\end{aligned} \tag{7.46}$$

having assumed that $\mathbf{c}^T\mathbf{s}^{(k)} \neq 0$. If $\mathbf{c} = \mathbf{s}^{(k)}$, the method employing (7.46) is known as the *symmetric Powell-Broyden method*. Denoting by B_{SPB} the corresponding matrix B_{k+1}, it can be shown that B_{SPB} is the unique solution to the problem:

$$\text{find } \bar{\mathrm{B}} \text{ such that } \|\bar{\mathrm{B}} - \mathrm{B}\|_F \text{ is minimized,}$$

where $\bar{\mathrm{B}}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}$ and $\|\cdot\|_F$ is the Frobenius norm.

As for the error made approximating $\mathrm{H}(\mathbf{x}^{(k+1)})$ with B_{SPB}, it can be proved that

$$\|\mathrm{B}_{SPB} - \mathrm{H}(\mathbf{x}^{(k+1)})\|_F \leq \|\mathrm{B}_k - \mathrm{H}(\mathbf{x}^{(k)})\|_F + 3L\|\mathbf{s}^{(k)}\|,$$

where it is assumed that H is Lipschitz continuous, with Lipschitz constant L, and that the iterates $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ belong to D.

To deal with the particular case in which the Hessian matrix is not only symmetric but also positive definite, we refer to [DS83], Section 9.2.

7.3 Constrained Optimization

The simplest case of constrained optimization can be formulated as follows. Given $f : \mathbb{R}^n \to \mathbb{R}$,

$$\text{minimize } f(\mathbf{x}), \text{ with } \mathbf{x} \in \Omega \subset \mathbb{R}^n. \tag{7.47}$$

More precisely, the point $\mathbf{x}^*$ is said to be a *global minimizer* in Ω if it satisfies (7.47), while it is a *local minimizer* if $\exists R > 0$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in B(\mathbf{x}^*; R) \subset \Omega.$$

Existence of solutions to problem (7.47) is, for instance, ensured by the Weierstrass theorem, in the case in which f is continuous and Ω is a closed and bounded set. Under the assumption that Ω is a convex set, the following optimality conditions hold.

Property 7.9 *Let $\Omega \subset \mathbb{R}^n$ be a convex set, $\mathbf{x}^* \in \Omega$ and $f \in C^1(B(\mathbf{x}^*; R))$, for a suitable $R > 0$. Then:*

1. *if $\mathbf{x}^*$ is a local minimizer of f then*

$$\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0, \forall \mathbf{x} \in \Omega; \tag{7.48}$$

2. *moreover, if f is convex on Ω (see (7.21)) and (7.48) is satisfied, then $\mathbf{x}^*$ is a global minimizer of f.*

We recall that $f : \Omega \to \mathbb{R}$ is a strongly convex function if $\exists \rho > 0$ such that

$$f[\alpha\mathbf{x} + (1-\alpha)\mathbf{y}] \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) - \alpha(1-\alpha)\rho\|\mathbf{x} - \mathbf{y}\|_2^2, \tag{7.49}$$

$\forall \mathbf{x}, \mathbf{y} \in \Omega$ and $\forall \alpha \in [0, 1]$. The following result holds.

Property 7.10 *Let $\Omega \subset \mathbb{R}^n$ be a closed and convex set and f be a strongly convex function in Ω. Then there exists a unique local minimizer $\mathbf{x}^* \in \Omega$.*

Throughout this section, we refer to [Avr76], [Ber82], [CCP70], [Lue73] and [Man69], for the proofs of the quoted results and further details.

A remarkable instance of (7.47) is the following problem: given $f : \mathbb{R}^n \to \mathbb{R}$,

$$\text{minimize } f(\mathbf{x}), \text{ under the constraint that } \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{7.50}$$

where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, is a given function of components $h_1, \ldots, h_m$. The analogues of critical points in problem (7.50) are called the *regular points.*

Definition 7.2 A point $\mathbf{x}^* \in \mathbb{R}^n$, such that $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$, is said to be *regular* if the column vectors of the Jacobian matrix $\mathrm{J}_{\mathbf{h}}(\mathbf{x}^*)$ are linearly independent, having assumed that $h_i \in C^1(B(\mathbf{x}^*; R))$, for a suitable $R > 0$ and $i = 1, \ldots, m$. ■

Our aim now is to convert problem (7.50) into an unconstrained minimization problem of the form (7.2), to which the methods introduced in Section 7.2 can be applied.

For this purpose, we introduce the *Lagrangian function* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}),$$

where the vector $\boldsymbol{\lambda}$ is called the *Lagrange multiplier.* Moreover, let us denote by $\mathrm{J}_{\mathcal{L}}$ the Jacobian matrix associated with $\mathcal{L}$, but where the partial derivatives are only taken with respect to the variables $x_1, \ldots, x_n$. The link between (7.2) and (7.50) is then expressed by the following result.

Property 7.11 *Let $\mathbf{x}^*$ be a local minimizer for* (7.50) *and suppose that, for a suitable $R > 0$, $f, h_i \in C^1(B(\mathbf{x}^*; R))$, for $i = 1, \ldots, m$. Then there exists a unique vector $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that $\mathrm{J}_{\mathcal{L}}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$.*

Conversely, assume that $\mathbf{x}^* \in \mathbb{R}^n$ satisfies $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$ and that, for a suitable $R > 0$ and $i = 1, \ldots, m$, $f, h_i \in C^2(B(\mathbf{x}^*; R))$. Let $\mathrm{H}_{\mathcal{L}}$ be the matrix of entries $\partial^2 \mathcal{L}/\partial x_i \partial x_j$ for $i, j = 1, \ldots, n$. If there exists a vector $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that $\mathrm{J}_{\mathcal{L}}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$ and

$$\mathbf{z}^T \mathrm{H}_{\mathcal{L}}(\mathbf{x}^*, \boldsymbol{\lambda}^*)\mathbf{z} > 0 \;\forall \mathbf{z} \neq \mathbf{0}, \;\; \textit{with} \;\; \nabla \mathbf{h}(\mathbf{x}^*)^T \mathbf{z} = 0,$$

then $\mathbf{x}^*$ is a strict local minimizer of (7.50)*.*

The last class of problems that we are going to deal with includes the case where inequality constraints are also present, i.e.: given $f : \mathbb{R}^n \to \mathbb{R}$,

$$\text{minimize } f(\mathbf{x}), \;\; \text{under the constraint that } \mathbf{h}(\mathbf{x}) = \mathbf{0} \text{ and } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \qquad (7.51)$$

where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^r$ are two given functions. It is understood that $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ means $g_i(\mathbf{x}) \leq 0$ for $i = 1, \ldots, r$. Inequality constraints give rise to some extra formal complication with respect to the case previously examined, but do not prevent converting the solution of (7.51) into the minimization of a suitable Lagrangian function.

In particular, Definition 7.2 becomes

Definition 7.3 Assume that $h_i, g_j \in C^1(B(\mathbf{x}^*; R))$ for a suitable $R > 0$ with $i = 1, \ldots, m$ and $j = 1, \ldots, r$, and denote by $\mathcal{J}(\mathbf{x}^*)$ the set of indices j such that $g_j(\mathbf{x}^*) = 0$. A point $\mathbf{x}^* \in \mathbb{R}^n$ such that $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{g}(\mathbf{x}^*) \leq \mathbf{0}$ is said to be *regular* if the column vectors of the Jacobian matrix $\mathrm{J}_{\mathbf{h}}(\mathbf{x}^*)$ together with the vectors $\nabla g_j(\mathbf{x}^*)$, $j \in \mathcal{J}(\mathbf{x}^*)$ form a set of linearly independent vectors. ■

Finally, an analogue of Property 7.11 holds, provided that the following Lagrangian function is used

$$\mathcal{M}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x})$$

instead of $\mathcal{L}$ and that further assumptions on the constraints are made.

For the sake of simplicity, we report in this case only the following necessary condition for optimality of problem (7.51) to hold.

Property 7.12 *Let $\mathbf{x}^*$ be a regular local minimizer for* (7.51) *and suppose that, for a suitable $R > 0$, $f, h_i, g_j \in C^1(B(\mathbf{x}^*; R))$ with $i = 1, \ldots, m$, $j = 1, \ldots, r$. Then, there exist only two vectors $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and $\boldsymbol{\mu}^* \in \mathbb{R}^r$, such that $\mathrm{J}_{\mathcal{M}}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{0}$ with $\mu_j^* \geq 0$ and $\mu_j^* g_j(\mathbf{x}^*) = 0 \; \forall j = 1, \ldots, r$.*

7.3.1 Kuhn-Tucker Necessary Conditions for Nonlinear Programming

In this section we recall some results, known as *Kuhn-Tucker conditions* [KT51], that ensure in general the existence of a local solution for the nonlinear programming problem. Under suitable assumptions they also guarantee the existence of a global solution. Throughout this section we suppose that a minimization problem can always be reformulated as a maximization one.

Let us consider the general nonlinear programming problem:

$$\begin{aligned}
&\text{given } f : \mathbb{R}^n \to \mathbb{R},\\
&\text{maximize } f(\mathbf{x}), \text{ subject to}\\
&g_i(\mathbf{x}) \le b_i, \quad i = 1, \dots, l,\\
&g_i(\mathbf{x}) \ge b_i, \quad i = l+1, \dots, k,\\
&g_i(\mathbf{x}) = b_i, \quad i = k+1, \dots, m,\\
&\mathbf{x} \ge \mathbf{0}.
\end{aligned} \tag{7.52}$$

A vector $\mathbf{x}$ that satisfies the constraints above is called a *feasible solution* of (7.52) and the set of the feasible solutions is called the *feasible region.* We assume henceforth that $f, g_i \in C^1(\mathbb{R}^n)$, $i = 1, \dots, m$, and define the sets $I_= = \{i : \ g_i(\mathbf{x}^*) = b_i\}$, $I_{\neq} = \{i : \ g_i(\mathbf{x}^*) \neq b_i\}$, $J_= = \{i : \ x_i^* = 0\}$, $J_> = \{i : \ x_i^* > 0\}$, having denoted by $\mathbf{x}^*$ a local maximizer of f. We associate with (7.52) the following Lagrangian

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \left[b_i - g_i(\mathbf{x})\right] - \sum_{i=m+1}^{m+n} \lambda_i x_{i-m}.$$

The following result can be proved.

Property 7.13 (Kuhn-Tucker conditions I and II) *If f has a constrained local maximum at the point $\mathbf{x} = \mathbf{x}^*$, it is necessary that a vector $\boldsymbol{\lambda}^* \in \mathbb{R}^{m+n}$ exists such that (first Kuhn-Tucker condition)*

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \le 0,$$

where strict equality holds for every component $i \in J_>$. Moreover (second Kuhn-Tucker condition)

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)^T \mathbf{x}^* = 0.$$

The other two necessary Kuhn-Tucker conditions are as follows.

Property 7.14 *Under the same hypothesis as in Property 7.13, the third Kuhn-Tucker condition requires that:*

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) &\geq 0 \quad && i = 1, \ldots, l, \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) &\leq 0 \quad && i = l+1, \ldots, k, \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) &= 0 \quad && i = k+1, \ldots, m. \end{aligned}$$

Moreover (fourth Kuhn-Tucker condition)

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)^T \mathbf{x}^* = 0.$$

It is worth noticing that the Kuhn-Tucker conditions hold provided that the vector $\boldsymbol{\lambda}^*$ exists. To ensure this, it is necessary to introduce a further geometric condition that is known as *constraint qualification* (see [Wal75], p. 48).

We conclude this section by the following fundamental theorem which establishes when the Kuhn-Tucker conditions become also sufficient for the existence of a global maximizer for f.

Property 7.15 *Assume that the function f in* (7.52) *is a concave function (i.e., $-f$ is convex) in the feasible region. Suppose also that the point* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ *satisfies all the Kuhn-Tucker necessary conditions and that the functions g_i for which $\lambda_i^* > 0$ are convex while those for which $\lambda_i^* < 0$ are concave. Then $f(\mathbf{x}^*)$ is the constrained global maximizer of f for problem* (7.52).

7.3.2 The Penalty Method

The basic idea of this method is to eliminate, partly or completely, the constraints in order to transform the constrained problem into an unconstrained one. This new problem is characterized by the presence of a parameter that yields a measure of the accuracy at which the constraint is actually imposed.

Let us consider the constrained problem (7.50), assuming we are searching for the solution $\mathbf{x}^*$ only in $\Omega \subset \mathbb{R}^n$. Suppose that such a problem admits at least one solution in Ω and write it in the following penalized form

$$\text{minimize } \mathcal{L}_\alpha(\mathbf{x}) \quad \text{for } \mathbf{x} \in \Omega, \tag{7.53}$$

where

$$\mathcal{L}_\alpha(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{2}\alpha \|\mathbf{h}(\mathbf{x})\|_2^2.$$

The function $\mathcal{L}_\alpha : \mathbb{R}^n \to \mathbb{R}$ is called the *penalized Lagrangian*, and α is called the *penalty parameter*. It is clear that if the constraint was exactly satisfied then minimizing f would be equivalent to minimizing $\mathcal{L}_\alpha$.

The penalty method is an iterative technique for solving (7.53).

For $k = 0, 1, \ldots,$ until convergence, one must solve the sequence of problems:

$$\text{minimize } \mathcal{L}_{\alpha_k}(\mathbf{x}) \quad \text{with } \mathbf{x} \in \Omega, \tag{7.54}$$

where $\{\alpha_k\}$ is an increasing monotonically sequence of positive penalty parameters, such that $\alpha_k \to \infty$ as $k \to \infty$. As a consequence, after choosing α_k, at each step of the penalty process we have to solve a minimization problem with respect to the variable $\mathbf{x}$, leading to a sequence of values $\mathbf{x}_k^*$, solutions to (7.54). By doing so, the objective function $\mathcal{L}_{\alpha_k}(\mathbf{x})$ tends to infinity, unless $\mathbf{h}(\mathbf{x})$ is equal to zero.

The minimization problems can then be solved by one of the methods introduced in Section 7.2. The following property ensures the convergence of the penalty method in the form (7.53).

Property 7.16 *Assume that $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, are continuous functions on a closed set $\Omega \subset \mathbb{R}^n$ and suppose that the sequence of penalty parameters $\alpha_k > 0$ is monotonically divergent. Finally, let $\mathbf{x}_k^*$ be the global minimizer of problem* (7.54) *at step k. Then, taking the limit as $k \to \infty$, the sequence $\mathbf{x}_k^*$ converges to $\mathbf{x}^*$, which is a global minimizer of f in Ω and satisfies the constraint $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$.*

Regarding the selection of the parameters α_k, it can be shown that large values of α_k make the minimization problem in (7.54) ill-conditioned, thus making its solution quite prohibitive unless the initial guess is particularly close to $\mathbf{x}^*$. On the other hand, the sequence α_k must not grow too slowly, since this would negatively affect the overall convergence of the method.

A choice that is commonly made in practice is to pick up a not too large value of α_0 and then set $\alpha_k = \beta\alpha_{k-1}$ for $k > 0$, where β is an integer number between 4 and 10 (see [Ber82]). Finally, the starting point for the numerical method used to solve the minimization problem (7.54) can be set equal to the last computed iterate.

The penalty method is implemented in Program 63. This requires as input parameters the functions `f`, `h`, an initial value `alpha0` for the penalty parameter and the number `beta`.

Program 63 - lagrpen : Penalty method

```
function [x,vinc,iter]=lagrpen(f,h,x0,h,tol,alpha0,beta)
%LAGRPEN Penalty method for constrained function optimization
% [X,VINC,ITER]=LAGRPEN(F,H,X0,TOL,ALPHA0,BETA) attempts to compute
% the minimizer X of a function F with the Penalty method. F is a string containing
% the functional expressions of the function. X0 specifies the initial guess. H is a
% string variable containing the constraint. TOL specifies the tolerance of the method.
% ALPHA0 and BETA are given parameters. ITER is the iteration number at which X is
% computed. VINC is the accuracy at which the constraint is satisfied.
x = x0; [r,c]=size(h); vinc = 0;
for i=1:r
   vinc = max(vinc,eval(h(i,1:c)));
end
norm2h=['(',h(1,1:c),')^2'];
for i=2:r
   norm2h=[norm2h,'+(',h(i,1:c),')^2'];
end
```

```
alpha = alpha0;
options(1)=0; options(2)=tol*0.1;
iter = 0;
while vinc > tol
   g=[f,'+0.5*',num2str(alpha,16),'*',norm2h];
   [x]=fmins(g,x,options);
   vinc=0;
   iter = iter + 1;
   for i=1:r
      vinc = max(vinc,eval(h(i,1:c)));
   end
   alpha=alpha*beta;
end
return
```

Example 7.6 Let us employ the penalty method to compute the minimizer of $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ under the constraint $h(\mathbf{x}) = (x_1 + 0.5)^2 + (x_2 + 0.5)^2 - 0.25 = 0$. The crosses in Figure 7.3 denote the sequence of iterates computed by Program 63 starting from $\mathbf{x}^{(0)} = [1, 1]^T$ and choosing $\alpha_0 = 0.1$, $\beta = 6$. The method converges in 12 iterations to the value $\mathbf{x} = [-0.2463, -0.0691]^T$, satisfying the constraint up to a tolerance of 10^{-4}. •

7.3.3 The Method of Lagrange Multipliers

A variant of the penalty method makes use of (instead of $\mathcal{L}_\alpha(\mathbf{x})$ in (7.53)) the *augmented Lagrangian* function $\mathcal{G}_\alpha : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ given by

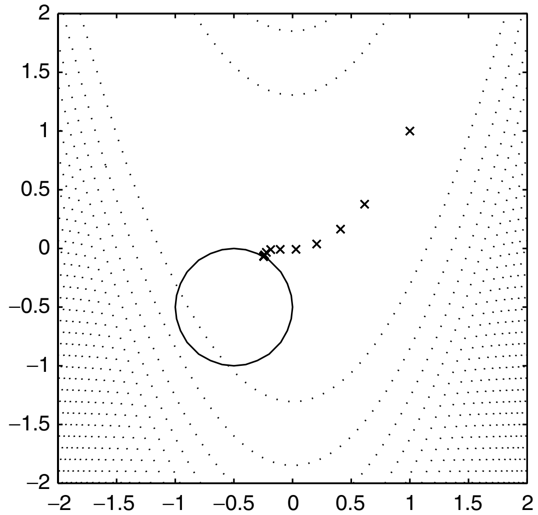

Fig. 7.3. Convergence history of the penalty method in Example 7.6

$$\mathcal{G}_\alpha(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T\mathbf{h}(\mathbf{x}) + \frac{1}{2}\alpha\|\mathbf{h}(\mathbf{x})\|_2^2, \qquad (7.55)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is a *Lagrange multiplier.* Clearly, if $\mathbf{x}^*$ is a solution to problem (7.50), then it will also be a solution to (7.55), but with the advantage, with respect to (7.53), of disposing of the further degree of freedom $\boldsymbol{\lambda}$. The penalty method applied to (7.55) reads: for $k = 0, 1, \ldots$, until convergence, solve the sequence of problems

$$\text{minimize } \mathcal{G}_{\alpha_k}(\mathbf{x},\boldsymbol{\lambda}_k) \quad \text{for } \mathbf{x} \in \Omega, \qquad (7.56)$$

where $\{\boldsymbol{\lambda}_k\}$ is a bounded sequence of unknown vectors in $\mathbb{R}^m$, and the parameters α_k are defined as above (notice that if $\boldsymbol{\lambda}_k$ were zero, then we would recover method (7.54)).

Property 7.16 also holds for method (7.56), provided that the multipliers are assumed to be bounded. Notice that the existence of the minimizer of (7.56) is not guaranteed, even in the case where f has a unique global minimizer (see Example 7.7). This circumstance can be overcome by adding further non quadratic terms to the augmented Lagrangian function (e.g., of the form $\|\mathbf{h}\|_2^p$, with p large).

Example 7.7 Let us find the minimizer of $f(x) = -x^4$ under the constraint $x = 0$. Such problem clearly admits the solution $x^* = 0$. If, instead, one considers the augmented Lagrangian function

$$\mathcal{L}_{\alpha_k}(x,\lambda_k) = -x^4 + \lambda_k x + \frac{1}{2}\alpha_k x^2,$$

one finds that it no longer admits a minimum at $x = 0$, though vanishing there, for any α_k different from zero. •

As far as the choice of the multipliers is concerned, the sequence of vectors $\boldsymbol{\lambda}_k$ is typically assigned by the following formula

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k\mathbf{h}(\mathbf{x}^{(k)}),$$

where $\boldsymbol{\lambda}_0$ is a given value while the sequence of α_k can be set a priori or modified during run-time.
As for the convergence properties of the method of Lagrange multipliers, the following local result holds.

Property 7.17 *Assume that* $\mathbf{x}^*$ *is a regular strict local minimizer of* (7.50) *and that:*

1. $f, h_i \in C^2(B(\mathbf{x}^*; R))$ *with* $i = 1, \ldots, m$ *and for a suitable* $R > 0$*;*
2. *the pair* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ *satisfies* $\mathbf{z}^T\mathrm{H}_{\mathcal{G}_0}(\mathbf{x}^*,\boldsymbol{\lambda}^*)\mathbf{z} > 0$, $\forall\mathbf{z} \neq \mathbf{0}$ *such that* $\mathrm{J}_\mathbf{h}(\mathbf{x}^*)^T\mathbf{z} = 0$*;*
3. $\exists\bar{\alpha} > 0$ *such that* $\mathrm{H}_{\mathcal{G}_{\bar{\alpha}}}(\mathbf{x}^*,\boldsymbol{\lambda}^*) > 0$.

Then, there exist three positive scalars δ, γ *and* M *such that, for any pair* $(\boldsymbol{\lambda}, \alpha) \in V = \left\{(\boldsymbol{\lambda}, \alpha) \in \mathbb{R}^{m+1} \ : \ \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|_2 < \delta\alpha, \ \alpha \geq \bar{\alpha}\right\}$*, the problem*

$$\textit{minimize } \mathcal{G}_\alpha(\mathbf{x}, \boldsymbol{\lambda}), \textit{ with } \mathbf{x} \in B(\mathbf{x}^*; \gamma),$$

admits a unique solution $\mathbf{x}(\boldsymbol{\lambda}, \alpha)$*, differentiable with respect to its arguments. Moreover,* $\forall(\boldsymbol{\lambda}, \alpha) \in V$

$$\|\mathbf{x}(\boldsymbol{\lambda}, \alpha) - \mathbf{x}^*\|_2 \leq M\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|_2.$$

Under further assumptions (see [Ber82], Proposition 2.7), it can be proved that the Lagrange multipliers method converges. Moreover, if $\alpha_k \to \infty$, as $k \to \infty$, then

$$\lim_{k\to\infty} \frac{\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}^*\|_2}{\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}^*\|_2} = 0$$

and the convergence of the method is more than linear.

In the case where the sequence α_k has an upper bound, the method converges linearly.

Finally, we notice that, unlike the penalty method, it is no longer necessary that the sequence of α_k tends to infinity. This, in turn, limits the ill-conditioning of problem (7.56) as α_k is growing. Another advantage concerns the convergence rate of the method, which turns out to be independent of the growth rate of the penalty parameter, in the case of the Lagrange multipliers technique. This of course implies a considerable reduction of the computational cost.

The method of Lagrange multipliers is implemented in Program 64. Compared with Program 63, this further requires in input the initial value `lambda0` of the multiplier.

Program 64 - lagrmult : Method of Lagrange multipliers

```
function [x,vinc,iter]=lagrmult(f,h,x0,lambda0,tol,alpha0,beta)
%LAGRMULT Method of Lagrange multipliers for constrained function optimization
% [X,VINC,ITER]=LAGRMULT(F,H,X0,LAMBDA0,TOL,ALPHA0,BETA) attempts
% to compute the minimizer X of a function F with the method of Lagrange
% multipliers. F ia a string containing the functional expressions of the function.
% X0 and LAMBDA0 specify the initial guesses. H is a string variable containing the
% constraint. TOL specifies the tolerance of the method. ALPHA0 and BETA are given
% parameters. ITER is the iteration number at which X is computed. VINC is the
% accuracy at which the constraint is satisfied.
x = x0; [r,c]=size(h); vinc = 0; lambda = lambda0;
for i=1:r
   vinc = max(vinc,eval(h(i,1:c)));
end
norm2h=['(',h(1,1:c),')^2'];
```

```
for i=2:r
    norm2h=[norm2h,'+(',h(i,1:c),')^2'];
end
alpha = alpha0;
options(1)=0; options(2)=tol*0.1;
iter = 0;
while vinc > tol
    lh=['(',h(1,1:c),')*',num2str(lambda(1))];
    for i=2:r
        lh=[lh,'+(',h(i,1:c),')*',num2str(lambda(i))];
    end
    g=[f,'+0.5*',num2str(alpha,16),'*',norm2h,'+',lh];
    [x]=fmins(g,x,options);
    vinc=0;
    iter = iter + 1;
    for i=1:r
        vinc = max(vinc,eval(h(i,1:c)));
    end
    alpha=alpha*beta;
    for i=1:r
        lambda(i)=lambda(i)+alpha*eval(h(i,1:c));
    end
end
return
```

Example 7.8 We use the method of Lagrange multipliers to solve the problem presented in Example 7.6. Set $\boldsymbol{\lambda} = 10$ and leave the remaining parameters unchanged. The method converges in 6 iterations and the crosses in Figure 7.4 show the iterates computed by Program 64. The constraint is here satisfied up to machine precision. •

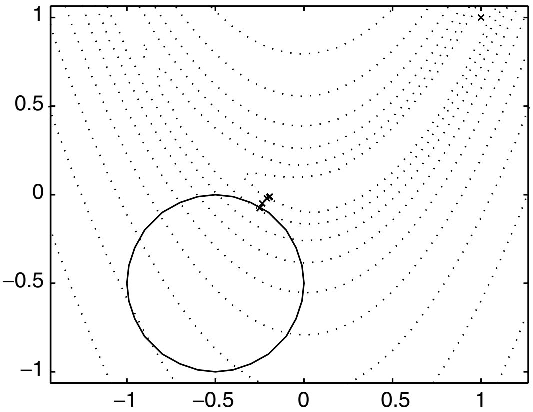

Fig. 7.4. Convergence history for the method of Lagrange multipliers in Example 7.8

7.4 Applications

The two applications of this section are concerned with nonlinear systems arising in the simulation of the electric potential in a semiconductor device and in the triangulation of a two-dimensional polygon.

7.4.1 Solution of a Nonlinear System Arising from Semiconductor Device Simulation

Let us consider the nonlinear system in the unknown $\mathbf{u} \in \mathbb{R}^n$

$$\mathbf{F}(\mathbf{u}) = \mathrm{A}\mathbf{u} + \boldsymbol{\phi}(\mathbf{u}) - \mathbf{b} = \mathbf{0}, \tag{7.57}$$

where $\mathrm{A} = (\lambda/h)^2 \mathrm{tridiag}_n(-1, 2-1)$, for $h = 1/(n+1)$, $\phi_i(\mathbf{u}) = 2K\sinh(u_i)$ for $i = 1, \ldots, n$, where λ and K are two positive constants and $\mathbf{b} \in \mathbb{R}^n$ is a given vector. Problem (7.57) arises in the numerical simulation of semiconductor devices in microelectronics, where $\mathbf{u}$ and $\mathbf{b}$ represent electric potential and doping profile, respectively.

In Figure 7.5 (*left*) we show schematically the particular device considered in the numerical example, a $p-n$ junction diode of unit normalized length, subject to an external bias $\triangle V = V_b - V_a$, together with the doping profile of the device, normalized to 1 (*right*). Notice that $b_i = b(x_i)$, for $i = 1, \ldots, n$, where $x_i = ih$. The mathematical model of the problem at hand comprises a nonlinear Poisson equation for the electric potential and two continuity equations of advection-diffusion type, as those addressed in Chapter 12, for the current densities. For the complete derivation of the model and its analysis see, for instance, [Mar86] and [Jer96].

Solving system (7.57) corresponds to finding the minimizer in $\mathbb{R}^n$ of the function $f : \mathbb{R}^n \to \mathbb{R}$ defined as

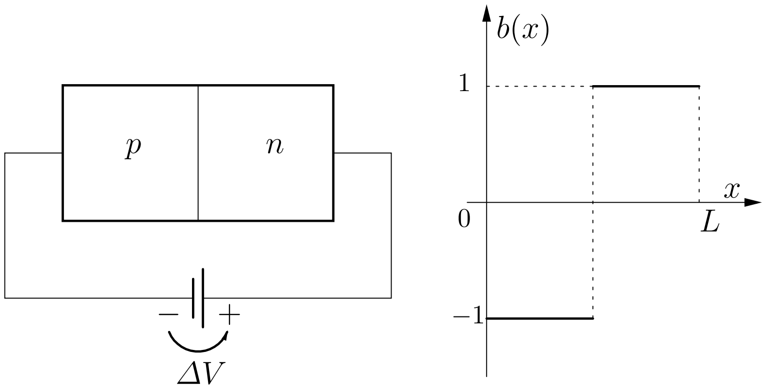

Fig. 7.5. Scheme of a semiconductor device (*left*); doping profile (*right*)

$$f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T \mathrm{A}\mathbf{u} + 2\sum_{i=1}^{n} \cosh(u_i) - \mathbf{b}^T\mathbf{u}. \tag{7.58}$$

It can be checked (see Exercise 5) that for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, with $\mathbf{u} \neq \mathbf{v}$, and for any $\lambda \in (0, 1)$

$$\lambda f(\mathbf{u}) + (1-\lambda)f(\mathbf{v}) - f(\lambda\mathbf{u} + (1-\lambda)\mathbf{v}) > (1/2)\lambda(1-\lambda)\|\mathbf{u} - \mathbf{v}\|_{\mathrm{A}}^2,$$

where $\|\cdot\|_{\mathrm{A}}$ denotes the energy norm introduced in (1.28). This implies that $f(\mathbf{u})$ is an uniformly convex function in $\mathbb{R}^n$, that is, it strictly satisfies (7.49) with $\rho = 1/2$.

Property 7.10 ensures, in turn, that the function in (7.58) admits a unique minimizer $\mathbf{u}^* \in \mathbb{R}^n$ and it can be shown (see Theorem 14.4.3, p. 503 [OR70]) that there exists a sequence $\{\alpha_k\}$ such that the iterates of the damped Newton method introduced in Section 7.2.6 converge to $\mathbf{u}^* \in \mathbb{R}^n$ (at least) superlinearly.

Thus, using the damped Newton method for solving system (7.57) leads to the following sequence of linearized problems: given $\mathbf{u}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \ldots$ until convergence solve

$$\left[\mathrm{A} + 2K\,\mathrm{diag}_n(\cosh(u_i^{(k)}))\right] \boldsymbol{\delta}\mathbf{u}^{(k)} = \mathbf{b} - \left(\mathrm{A}\mathbf{u}^{(k)} + \boldsymbol{\phi}(\mathbf{u}^{(k)})\right), \tag{7.59}$$

then set $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \alpha_k \boldsymbol{\delta}\mathbf{u}^{(k)}$.

Let us now address two possible choices of the acceleration parameters α_k. The first one has been proposed in [BR81] and is

$$\alpha_k = \frac{1}{1 + \rho_k\, \|\mathbf{F}(\mathbf{u}^{(k)})\|_\infty}, \qquad k = 0, 1, \ldots, \tag{7.60}$$

where the coefficients $\rho_k \geq 0$ are suitable acceleration parameters picked in such a way that the descent condition $\|\mathbf{F}(\mathbf{u}^{(k)} + \alpha_k \delta\mathbf{u}^{(k)})\|_\infty < \|\mathbf{F}(\mathbf{u}^{(k)})\|_\infty$ is satisfied (see [BR81] for the implementation details of the algorithm).

We notice that, as $\|\mathbf{F}(\mathbf{u}^{(k)})\|_\infty \to 0$, (7.60) yields $\alpha_k \to 1$, thus recovering the full (quadratic) convergence of Newton's method. Otherwise, as typically happens in the first iterations, $\|\mathbf{F}(\mathbf{u}^{(k)})\|_\infty \gg 1$ and α_k is quite close to zero, with a strong reduction of the Newton variation (damping).

As an alternative to (7.60), the sequence $\{\alpha_k\}$ can be generated using the simpler formula, suggested in [Sel84], Chapter 7

$$\alpha_k = 2^{-i(i-1)/2}, \qquad k = 0, 1, \ldots, \tag{7.61}$$

where i is the first integer in the interval $[1, It_{max}]$ such that the descent condition above is satisfied, It_{max} being the maximum admissible number of damping cycles for any Newton's iteration (fixed equal to 10 in the numerical experiments).

As a comparison, both damped and standard Newton's methods have been implemented, the former one with both choices (7.60) and (7.61) for the coefficients α_k. In the case of Newton's method, we have set in (7.59) $\alpha_k = 1$ for any $k \geq 0$.

The numerical examples have been performed with $n = 49$, $b_i = -1$ for $i \leq n/2$ and the remaining values b_i equal to 1. Moreover, we have taken $\lambda^2 = 1.67 \cdot 10^{-4}$, $K = 6.77 \cdot 10^{-6}$ and fixed the first $n/2$ components of the initial vector $\mathbf{u}^{(0)}$ equal to V_a and the remaining ones equal to V_b, where $V_a = 0$ and $V_b = 10$.

The tolerance on the maximum change between two successive iterates, which monitors the convergence of damped Newton's method (7.59), has been set equal to 10^{-4}.

Figure 7.6 (*left*) shows the log-scale absolute error for the three algorithms as functions of the iteration number. Notice the rapid convergence of the damped Newton's method (8 and 10 iterations in the case of (7.60) and (7.61), respectively), compared with the extremely slow convergence of the standard Newton's method (192 iterations). Moreover, it is interesting to analyze in Figure 7.6 (*right*) the plot of the sequences of parameters α_k as functions of the iteration number.

The starred and the circled curves refer to the choices (7.60) and (7.61) for the coefficients α_k, respectively. As previously observed, the α_k's start from very small values, to converge quickly to 1 as the damped Newton method (7.59) enters the attraction region of the minimizer $\mathbf{x}^*$.

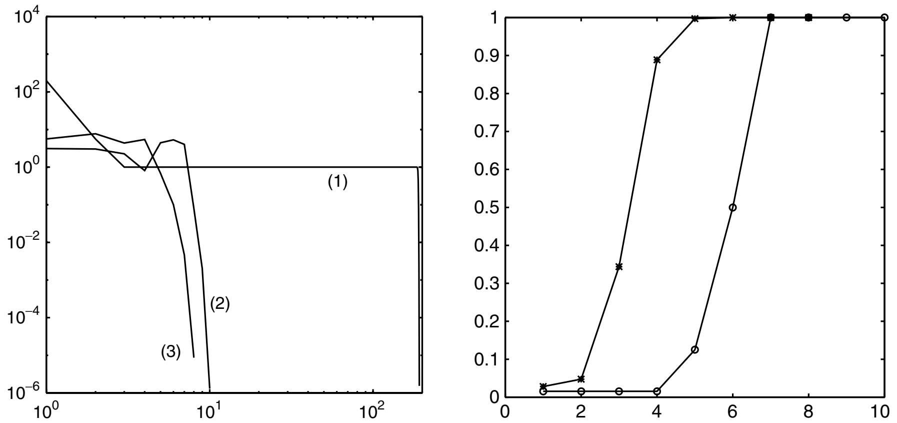

Fig. 7.6. Absolute error (*left*) and damping parameters α_k (*right*). The error curve for standard Newton's method is denoted by (1), while (2) and (3) refer to damped Newton's method with the choices (7.61) and (7.60) for the coefficients α_k, respectively

7.4.2 Nonlinear Regularization of a Discretization Grid

In this section we go back to the problem of regularizing a discretization grid that has been introduced in Section 3.14.2. There, we considered the technique of barycentric regularization, which leads to solving a linear system, typically of large size and featuring a sparse coefficient matrix.

In this section we address two alternative techniques, denoted as regularization *by edges* and *by areas*. The main difference with respect to the method described in Section 3.14.2 lies in the fact that these new approaches lead to systems of *nonlinear* equations.

Using the notation of Section 3.14.2, for each pair of nodes $\mathbf{x}_j$, $\mathbf{x}_k \in \mathcal{Z}_i$, denote by l_{jk} the edge on the boundary $\partial\mathcal{P}_i$ of $\mathcal{P}_i$ which connects them and by $\mathbf{x}_{jk}$ the midpoint of l_{jk}, while for each triangle $T \in \mathcal{P}_i$ we denote by $\mathbf{x}_{b,T}$ the centroid of T. Moreover, let $n_i = \dim(\mathcal{Z}_i)$ and denote for any geometric entity (side or triangle) by $|\cdot|$ its measure in $\mathbb{R}^1$ or $\mathbb{R}^2$.

In the case of regularization by edges, we let

$$\mathbf{x}_i = \left(\sum_{l_{jk}\in\partial\mathcal{P}_i} \mathbf{x}_{jk}|l_{jk}| \right) /|\partial\mathcal{P}_i|, \qquad \forall \mathbf{x}_i \in \mathcal{N}_h, \tag{7.62}$$

while in the case of regularization by areas, we let

$$\mathbf{x}_i = \left(\sum_{T\in\mathcal{P}_i} \mathbf{x}_{b,T}|T| \right) /|\mathcal{P}_i|, \qquad \forall \mathbf{x}_i \in \mathcal{N}_h. \tag{7.63}$$

In both the regularization procedures we assume that $\mathbf{x}_i = \mathbf{x}_i^{(\partial D)}$ if $\mathbf{x}_i \in \partial D$, that is, the nodes lying on the boundary of the domain D are fixed. Letting $n = N - N_b$ be the number of internal nodes, relation (7.62) amounts to solving the following two systems of nonlinear equations for the coordinates $\{x_i\}$ and $\{y_i\}$ of the internal nodes, with $i = 1, \ldots, n$

$$\begin{aligned} x_i - \frac{1}{2}\left(\sum_{l_{jk}\in\partial\mathcal{P}_i} (x_j + x_k)|l_{jk}| \right) / \sum_{l_{jk}\in\partial\mathcal{P}_i} |l_{jk}| = 0, \\ y_i - \frac{1}{2}\left(\sum_{l_{jk}\in\partial\mathcal{P}_i} (y_j + y_k)|l_{jk}| \right) / \sum_{l_{jk}\in\partial\mathcal{P}_i} |l_{jk}| = 0. \end{aligned} \tag{7.64}$$

Similarly, (7.63) leads to the following nonlinear systems, for $i = 1, \ldots, n$

$$\begin{aligned} x_i - \frac{1}{3}\left(\sum_{T\in\mathcal{P}_i} (x_{1,T} + x_{2,T} + x_{3,T})|T| \right) / \sum_{T\in\mathcal{P}_i} |T| = 0, \\ y_i - \frac{1}{3}\left(\sum_{T\in\mathcal{P}_i} (y_{1,T} + y_{2,T} + y_{3,T})|T| \right) / \sum_{T\in\mathcal{P}_i} |T| = 0, \end{aligned} \tag{7.65}$$

where $\mathbf{x}_{s,T} = [x_{s,T}, y_{s,T}]^T$, for $s = 1, 2, 3$, are the coordinates of the vertices of each triangle $T \in \mathcal{P}_i$. Notice that the nonlinearity of systems (7.64) and (7.65) is due to the presence of terms $|l_{jk}|$ and $|T|$.

Both systems (7.64) and (7.65) can be cast in the form (7.1), denoting, as usual, by f_i the i-th nonlinear equation of the system, for $i = 1, \ldots, n$. The complex functional dependence of f_i on the unknowns makes it prohibitive to use Newton's method (7.4), which would require the explicit computation of the Jacobian matrix $\mathrm{J}_{\mathbf{F}}$.

A convenient alternative is provided by the *nonlinear Gauss-Seidel method* (see [OR70], Chapter 7), which generalizes the corresponding method proposed in Chapter 4 for linear systems and can be formulated as follows.

Denote by z_i, for $i = 1, \ldots, n$, either of the unknown x_i or y_i. Given the initial vector $\mathbf{z}^{(0)} = [z_1^{(0)}, \ldots, z_n^{(0)}]^T$, for $k = 0, 1, \ldots$ until convergence, solve

$$f_i(z_1^{(k+1)}, \ldots, z_{i-1}^{(k+1)}, \xi, z_{i+1}^{(k)}, \ldots, z_n^{(k)}) = 0, \qquad i = 1, \ldots, n, \tag{7.66}$$

then, set $z_i^{(k+1)} = \xi$. Thus, the nonlinear Gauss-Seidel method converts problem (7.1) into the successive solution of n scalar nonlinear equations. In the case of system (7.64), each of these equations is *linear* in the unknown $z_i^{(k+1)}$ (since ξ does not explicitly appear in the bracketed term at the right side of (7.64)). This allows for its exact solution in one step.

In the case of system (7.65), the equation (7.66) is genuinely nonlinear with respect to ξ, and is solved taking one step of a fixed-point iteration.

The nonlinear Gauss-Seidel (7.66) has been implemented in MATLAB to solve systems (7.64) and (7.65) in the case of the initial triangulation shown in Figure 7.7 (*left*). Such a triangulation covers the external region of a two dimensional wing section of type NACA 2316. The grid contains $N_T = 534$ triangles and $n = 198$ internal nodes.

The algorithm reached convergence in 42 iterations for both kinds of regularization, having used as stopping criterion the test $\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\|_\infty \leq 10^{-4}$. In Figure 7.7 (*right*) the discretization grid obtained after the regularization

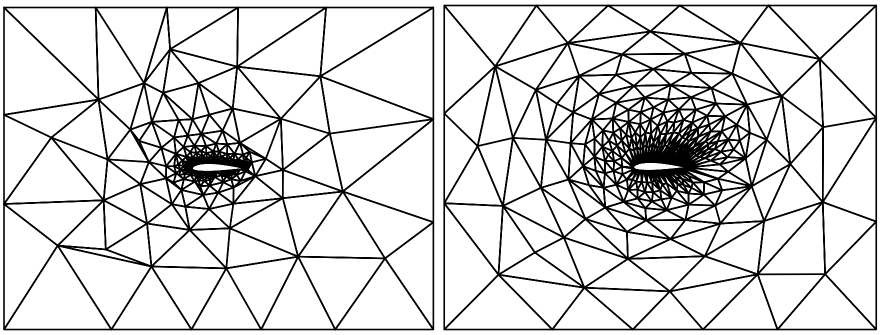

Fig. 7.7. Triangulation before (*left*) and after (*right*) the regularization

by areas is shown (a similar result has been provided by the regularization by edges). Notice the higher uniformity of the triangles with respect to those of the starting grid.

7.5 Exercises

1. Prove (7.8) for the m-step Newton-SOR method.
[*Hint*: use the SOR method for solving a linear system A$\mathbf{x}$=$\mathbf{b}$ with A=D-E-F and express the k-th iterate as a function of the initial datum $\mathbf{x}^{(0)}$, obtaining
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(0)} + (\mathrm{M}^{k+1} - \mathrm{I})\mathbf{x}^{(0)} + (\mathrm{M}^{k} + \ldots + \mathrm{I})\mathrm{B}^{-1}\mathbf{b},$$
where B= ω^{-1}(D$-\omega$E) and M $= \mathrm{B}^{-1}\omega^{-1}\left[(1-\omega)\mathrm{D} + \omega F\right]$. Since $\mathrm{B}^{-1}\mathrm{A} = \mathrm{I} - \mathrm{M}$ and
$$(\mathrm{I} + \ldots + \mathrm{M}^{k})(\mathrm{I} - \mathrm{M}) = \mathrm{I} - \mathrm{M}^{k+1}$$
then (7.8) follows by suitably identifying the matrix and the right-side of the system.]
2. Prove that using the gradient method for minimizing $f(x) = x^2$ with the directions $p^{(k)} = -1$ and the parameters $\alpha_k = 2^{-k+1}$, does not yield the minimizer of f.
3. Show that for the *steepest descent* method applied to minimizing a quadratic functional f of the form (7.35) the following inequality holds
$$f(\mathbf{x}^{(k+1)}) \le \left(\frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}\right)^2 f(\mathbf{x}^{(k)}),$$
where $\lambda_{max}, \lambda_{min}$ are the eigenvalues of maximum and minimum module, respectively, of the matrix A that appears in (7.35).
[*Hint*: proceed as done for (7.38).]
4. Check that the parameters α_k of Exercise 2 do not fulfill the conditions (7.31) and (7.32).
5. Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ introduced in (7.58) and check that it is uniformly convex on $\mathbb{R}^n$, that is
$$\lambda f(\mathbf{u}) + (1-\lambda) f(\mathbf{v}) - f(\lambda\mathbf{u} + (1-\lambda)\mathbf{v}) > (1/2)\lambda(1-\lambda)\|\mathbf{u} - \mathbf{v}\|_{\mathrm{A}}^2$$
for any $\mathbf{u}$, $\mathbf{v} \in \mathbb{R}^n$ with $\mathbf{u} \neq \mathbf{v}$ and $0 < \lambda < 1$.
[*Hint*: notice that cosh($\cdot$) is a convex function.]
6. To solve the nonlinear system
$$\begin{cases} -\dfrac{1}{81}\cos x_1 + \dfrac{1}{9}x_2^2 + \dfrac{1}{3}\sin x_3 = x_1, \\ \dfrac{1}{3}\sin x_1 + \dfrac{1}{3}\cos x_3 = x_2, \\ -\dfrac{1}{9}\cos x_1 + \dfrac{1}{3}x_2 + \dfrac{1}{6}\sin x_3 = x_3, \end{cases}$$
use the fixed-point iteration $\mathbf{x}^{(n+1)} = \Psi(\mathbf{x}^{(n)})$, where $\mathbf{x} = [x_1, x_2, x_3]^T$ and $\Psi(\mathbf{x})$ is the left-hand side of the system. Analyze the convergence of the iteration to compute the fixed point $\boldsymbol{\alpha} = [0, 1/3, 0]^T$.
[*Solution*: the fixed-point method is convergent since $\|\Psi(\boldsymbol{\alpha})\|_\infty = 1/2$.]

7. Using Program 50 implementing Newton's method, determine the global maximizer of the function

$$f(x) = e^{-\frac{x^2}{2}} - \frac{1}{4}\cos(2x)$$

and analyze the performance of the method (input data: `xv=1; tol=1e-6; nmax=500`). Solve the same problem using the following fixed-point iteration

$$x_{(k+1)} = g(x_k) \qquad \text{with } g(x) = \sin(2x)\left[\frac{e^{\frac{x^2}{2}}\,(x\sin(2x) + 2\cos(2x)) - 2}{2\,(x\sin(2x) + 2\cos(2x))}\right].$$

Analyze the performance of this second scheme, both theoretically and experimentally, and compare the results obtained using the two methods.
[*Solution*: the function f has a global maximum at $x = 0$. This point is a double zero for f'. Thus, Newton's method is only linearly convergent. Conversely, the proposed fixed-point method is third-order convergent.]

Springer

http://www.springer.com/978-3-540-34658-6

Numerical Mathematics
Quarteroni, A.; Sacco, R.; Saleri, F.
2007, XVIII, 657 p. 135 illus., Hardcover
ISBN: 978-3-540-34658-6