10. Function Minimization

Locating extreme values (maxima and minima) is particularly important in data analysis. This task occurs in solving the least-squares problem in the form $M(\mathbf{x}, \mathbf{y}) = \min$ and in the maximum likelihood problem as $L = \max$. By means of a simple change of sign, the latter problem can also be treated as locating a minimum. We always speak therefore of *minimization*.

10.1 Overview: Numerical Accuracy

We consider first the simple *quadratic form*

$$M(x) = c - bx + \frac{1}{2}Ax^2 \quad . \tag{10.1.1}$$

It has an extremum where the first derivative vanishes,

$$\frac{\mathrm{d}M}{\mathrm{d}x} = 0 = -b + Ax \quad , \tag{10.1.2}$$

that is, at the value

$$x_\mathrm{m} = \frac{b}{A} \quad . \tag{10.1.3}$$

With $M(x_\mathrm{m}) = M_\mathrm{m}$, Eq. (10.1.1) can easily be put into the form

$$M(x) - M_\mathrm{m} = \frac{1}{2}A(x - x_\mathrm{m})^2 \quad . \tag{10.1.4}$$

Although the function whose minimum we want to find does not usually have the simple form of (10.1.1), it can nevertheless be approximated by a quadratic form in the region of the minimum, where one has the Taylor expansion around the point x_0,

S. Brandt, *Data Analysis: Statistical and Computational Methods for Scientists and Engineers*,
DOI 10.1007/978-3-319-03762-2_10, © Springer International Publishing Switzerland 2014

$$M(x) = M(x_0) - b(x - x_0) + \frac{1}{2}A(x - x_0)^2 + \cdots \quad , \tag{10.1.5}$$

where

$$b = -M'(x_0) \quad , \qquad A = M''(x_0) \quad . \tag{10.1.6}$$

In this approximation the minimum is given by the point where the derivative $M'(x)$ is zero, i.e.,

$$x_{\mathrm{mp}} = x_0 + \frac{b}{A} \quad . \tag{10.1.7}$$

This holds only if x_0 is sufficiently close to the minimum so that terms of order higher than quadratic in (10.1.5) can be neglected. The situation is depicted in Fig. 10.1. The function $M(x)$ has a minimum at x_{m}, maxima at x_{M}, and points of inflection at x_{s}. For the second derivative in the region $x > x_{\mathrm{m}}$ one has $M''(x) > 0$ for $x < x_{\mathrm{s}}$ and $M(x'') < 0$ for $x > x_{\mathrm{s}}$. If we now choose x_0 to be in the region $x_{\mathrm{m}} < x_0 < x_{\mathrm{M}}$, then the first derivative $M'(x)$ there is always positive. Therefore x_{mp} lies closer to x_{m} only if $x_0 < x_{\mathrm{s}}$. Clearly the point x_0 is not in general chosen arbitrarily, but rather as close as possible to where the minimum is expected to be. We can call this estimated value the *zeroth approximation* of x_{m}. Various strategies are available to obtain successively better approximations:

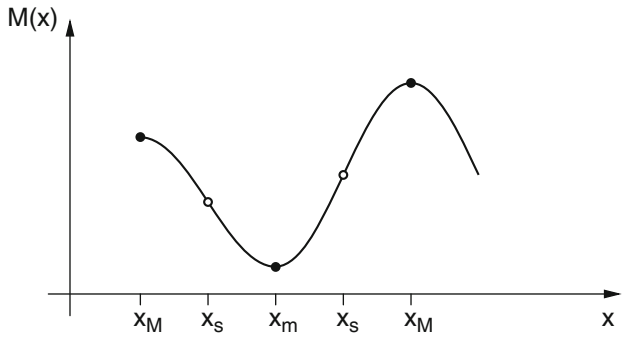

Fig. 10.1: The function $M(x)$ has a minimum at x_{m}, maxima at the points x_{M}, and points of inflection at x_{s}.

(i) *Use of the function and its first and second derivatives at* x_0.

One computes x_{mp} according to (10.1.7), takes x_{mp} as a first approximation, i.e., x_0 is replaced by x_{mp}, and obtains by repeated application of (10.1.7) a second approximation. The procedure is repeated until two successive approximations differ by less than a given value ε. From the discussion above it follows that this procedure does not converge if the zeroth approximation lies outside the points of inflection in Fig. 10.1.

(ii) *Use of the function and its first derivative at* x_0.

The sign of the derivative $M'(x_0) = -b$ at the point x_0 determines the direction in which the function increases. It is assumed that one

should search for the minimum in the direction in which the function decreases. The quantity

$$x_1 = x_0 + b \tag{10.1.8}$$

is computed, i.e., one replaces the second derivative by the value unity.

Instead of x_0 one now uses x_1 as the approximation, and so forth. Alternatively one can also use instead of (10.1.8) the rule

$$x_1 = x_0 + cb \quad , \tag{10.1.9}$$

where c is an arbitrary positive constant. Both rules ensure that the step from x_0 to x_1 proceeds in the direction of the minimum. If in addition one chooses c to be small, then the step is small, so that it does not go beyond (or not far beyond) the minimum.

(iii) *Use of the function at various points.*

The procedure (i) can be carried out without knowing the derivative of the function if the function itself is known at three points. One can then uniquely fit a parabola through these three points and take the extreme value as an approximation of the minimum of the function. One speaks of locating the minimum by *quadratic interpolation*. It is, however, by no means certain that the extreme value of the parabola is a minimum and not a maximum. As in procedure (i) it is therefore important that the three chosen points are already in the region of the minimum of the function.

(iv) *Successive reduction of an interval containing the minimum.*

In none of the procedures discussed up to this point were we able to guaranty that the minimum of the function would actually be found. The minimum can be found with certainty provided one knows an interval $x_a < x < x_b$ containing the minimum. If such an interval is known, one can locate the minimum with arbitrary accuracy by successively subdividing and checking in which subinterval the minimum is located.

In Sects. 10.2–10.7 we shall examine the minimization of a function of only one variable. In Sect. 10.2 the formula for a parabola determined by three points will be given. In Sect. 10.3 it is shown that the minimization of a function of one variable is equivalent to the minimization of a function of n variables on a line in an n-dimensional space. Section 10.4 describes a procedure for locating an interval containing the minimum. In Sect. 10.5 a minimum search by means of interval division is described. This is combined in Sect. 10.5 with the procedure of quadratic interpolation in a way that the

interpolation is only used when it leads quickly to the minimum of the function. If this is not the case, one continues with interval division. In this way we possess a procedure with the certainty of interval division combined with – as much as possible – the speed of quadratic interpolation. The same procedure can also be used for a function of n variables if the search for the minimum is restricted to a line in the n-dimensional space. This problem is addressed in Sect. 10.7.

Next we turn to the task of searching for the minimum of a function of n variables. We begin in Sect. 10.8 with the particularly elegant simplex method. This is followed by the discussion of various procedures of successive minimization along fixed directions in the n-dimensional space. These directions can simply be the directions of the coordinates (Sect. 10.9), or for a function that depends only quadratically on the variables they can be chosen such that the minimum is reached in at most n steps (Sects. 10.10 and 10.11).

Finally we discuss a procedure of n-dimensional minimization which employs aspects of methods (i) and (ii) of the one-dimensional case. If $\mathbf{x}$ is the n-dimensional vector of the variables, then the general quadratic form, i.e., the generalization of (10.1.1) to n variables, is

$$\begin{aligned} M(\mathbf{x}) &= c - \mathbf{b}\cdot\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}} A\mathbf{x} \\ &= c - \sum_k b_k x_k + \frac{1}{2}\sum_{k,\ell} x_k A_{k\ell} x_\ell \quad . \end{aligned} \tag{10.1.10}$$

Here A is a symmetric matrix, $A_{k\ell} = A_{\ell k}$. The partial derivative with respect to x_i is

$$\begin{aligned} \frac{\partial M}{\partial x_i} &= -b_i + \frac{1}{2}\left(\sum_\ell A_{i\ell} x_\ell + \sum_k x_k A_{ki}\right) \\ &= -b_i + \sum_\ell A_{i\ell} x_\ell \quad . \end{aligned} \tag{10.1.11}$$

Expressing all of the partial derivatives as a vector ∇M gives

$$\nabla M = -\mathbf{b} + A\mathbf{x} \quad . \tag{10.1.12}$$

At the minimum point the vector of derivatives vanishes. The minimum is therefore located at

$$\mathbf{x}_{\mathrm{m}} = A^{-1}\mathbf{b} \quad , \tag{10.1.13}$$

in analogy to (10.1.3).

Clearly the function $M(\mathbf{x})$ does not in general have the simple form of (10.1.10). We can, however, expand it in a series around the point $\mathbf{x}_0$,

$$M(\mathbf{x}) = M(\mathbf{x}_0) - \mathbf{b}(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}} A(\mathbf{x}-\mathbf{x}_0) + \cdots \quad , \tag{10.1.14}$$

with the negative *gradient*

$$\mathbf{b} = -\nabla M(\mathbf{x}_0) \quad , \qquad \text{i.e.,} \qquad b_i = -\left.\frac{\partial M}{\partial x_i}\right|_{\mathbf{x}=\mathbf{x}_0} \quad , \tag{10.1.15}$$

and the *Hessian matrix* of second derivatives

$$A_{ik} = \left.\frac{\partial^2 M}{\partial x_i \partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \quad . \tag{10.1.16}$$

The series (10.1.14) is the starting point for various minimization procedures:

(i) *Minimization in the direction of the gradient.*

Starting from the point $\mathbf{x}_0$ one searches for the minimum along the direction given by the gradient $\nabla M(\mathbf{x}_0)$ and calls the point where it is found $\mathbf{x}_1$. Starting from $\mathbf{x}_1$ one looks for the minimum along the direction $\nabla M(\mathbf{x}_1)$, and so forth. We will discuss this procedure, called *minimization in the direction of steepest descent*, in Sect. 10.12.

(ii) *Step of given size in the gradient direction.*

One computes in analogy to (10.1.9)

$$\mathbf{x}_1 = \mathbf{x}_0 + c\mathbf{b} \quad , \qquad \mathbf{b} = -\nabla M(\mathbf{x}_0) \quad , \tag{10.1.17}$$

with a given positive c. That is, one takes a *step in the direction of steepest descent* of the function, without, however, searching exactly for the minimum in this direction. Next one computes the gradient at $\mathbf{x}_1$, steps from $\mathbf{x}_1$ in the direction of this gradient, etc. In Sect. 10.13 we will combine this method with the following one.

(iii) *Use of the gradient and the Hessian matrix at* $\mathbf{x}_0$.

If we truncate (10.1.14) after the quadratic term, we obtain a function whose minimum is, according to (10.1.13), given by

$$\mathbf{x}_{\mathrm{mp}} = \mathbf{x}_0 + A^{-1}\mathbf{b} \quad . \tag{10.1.18}$$

We take $\mathbf{x}_1 = \mathbf{x}_{\mathrm{mp}}$ as the first approximation, compute for this point the gradient and Hessian matrix, obtain by corresponding use of (10.1.18) the second approximation, and so forth. This procedure is discussed

in Sect. 10.14. It converges quickly if the zeroth approximation $\mathbf{x}_0$ is sufficiently close to the minimum. If that is not the case, however, then it gives – as for the corresponding one dimensional procedure – no reasonable solution. We will combine it, therefore, in Sect. 10.15 with method (ii), in order to obtain, when possible, the speed of (iii), but when necessary, the certainty of (ii).

In Sects. 10.8 through 10.15 very different methods for solving the same problem, the minimization of a function of n variables, will be discussed. In Sect. 10.16 we give information on how to choose one of the methods appropriate for the problem in question. Section 10.17 is dedicated to considerations of errors. In Sect. 10.18 several examples are discussed in detail.

Before we find the minimum x_m of a function, we would like to inquire briefly about the numerical accuracy we expect for x_m. The minimum is after all almost always determined by a comparison of values of the function at points close in x. If we solve (10.1.4) for $(x - x_\mathrm{m})$, we obtain

$$(x - x_\mathrm{m}) = \sqrt{\frac{2[M(x) - M(x_\mathrm{m})]}{A}} \quad .$$

We assume that A, i.e., the second derivative of the function M, is of order of magnitude unity close to the minimum. (This need only be true approximately. In fact, in numerical calculations one always scales all of the quantities such that they are of order unity, i.e., not something like 10^6 or 10^{-6}.) If we compute the function M with the precision δ then the difference $M(x) - M(x_\mathrm{m})$ is also known at best with a precision of δ, i.e., two function values can not be considered as being significantly different if they differ by only δ. For the corresponding x values one then has

$$(x - x_\mathrm{m}) \approx \sqrt{\frac{2\delta}{A}} \approx \sqrt{\delta} \quad . \tag{10.1.19}$$

If the computer has n binary places available for representing the mantissa, then a value x can be represented with the precision (4.2.7)

$$\frac{\Delta x}{x} = 2^{-n} \quad .$$

For computing a value x one chooses therefore a relative precision

$$\varepsilon \geq 2^{-n} \quad ,$$

since it is clearly pointless to try to compute a value with a higher precision than that with which it can be represented. If x is computed iteratively, i.e., one

computes a series $x_0, x_1, \ldots$ of approximations for x, then one can truncate this series as soon as

$$\frac{|x_k - x_{k-1}|}{|x_k|} < \varepsilon$$

or

$$|x_k - x_{k-1}| < \varepsilon |x_k| \tag{10.1.20}$$

for a given ε. With this prescription we will have difficulties, however, if $x_k = 0$. We introduce, therefore, in addition to ε a constant $t \neq 0$ and extend (10.1.20) to

$$|x_k - x_{k-1}| < \varepsilon |x_k| + t \quad . \tag{10.1.21}$$

The last task remaining is to choose the numerical values for ε and t. If x is the position of the minimum, then by (10.1.19) a value for ε must be chosen that is greater than or equal to the square root of the relative precision for the representation of a floating point number. With computations using "double precision" in Java there are $n = 53$ binary places available for the representation of the mantissa. Then only the values

$$\varepsilon > 2^{-n/2} \approx 2 \cdot 10^{-8}$$

are reasonable. The quantity t corresponds to an absolute precision. Therefore it can be chosen to be considerably smaller.

10.2 Parabola Through Three Points

If three points $(x_a, y_a), (x_b, y_b), (x_c, y_c)$ of a function are known, then we can determine the parabola

$$y = a_0 + a_1 x + a_2 x^2 \tag{10.2.1}$$

that passes through these points. Instead of (10.2.1) we can also represent the parabola by

$$y = c_0 + c_1 (x - x_b) + c_2 (x - x_b)^2 \quad . \tag{10.2.2}$$

This relationship is naturally valid for three given points, i.e.,

$$y_b = c_0$$

and

$$(y_a - y_b) = c_1 (x_a - x_b) + c_2 (x_a - x_b)^2 \quad ,$$

$$(y_c - y_b) = c_1 (x_c - x_b) + c_2 (x_c - x_b)^2 \quad .$$

From this we obtain

$$c_1 = C[(x_c - x_b)^2(y_a - y_b) - (x_a - x_b)^2(y_c - y_b)] \quad , \tag{10.2.3}$$

$$c_2 = C[-(x_c - x_b)(y_a - y_b) + (x_a - x_b)(y_c - y_b)] \tag{10.2.4}$$

with

$$C = \frac{1}{(x_a - x_b)(x_c - x_b)^2 - (x_c - x_b)(x_a - x_b)^2}$$

and for the extremum of the parabola

$$x_{\mathrm{mp}} = x_b - \frac{c_1}{2c_2} \quad . \tag{10.2.5}$$

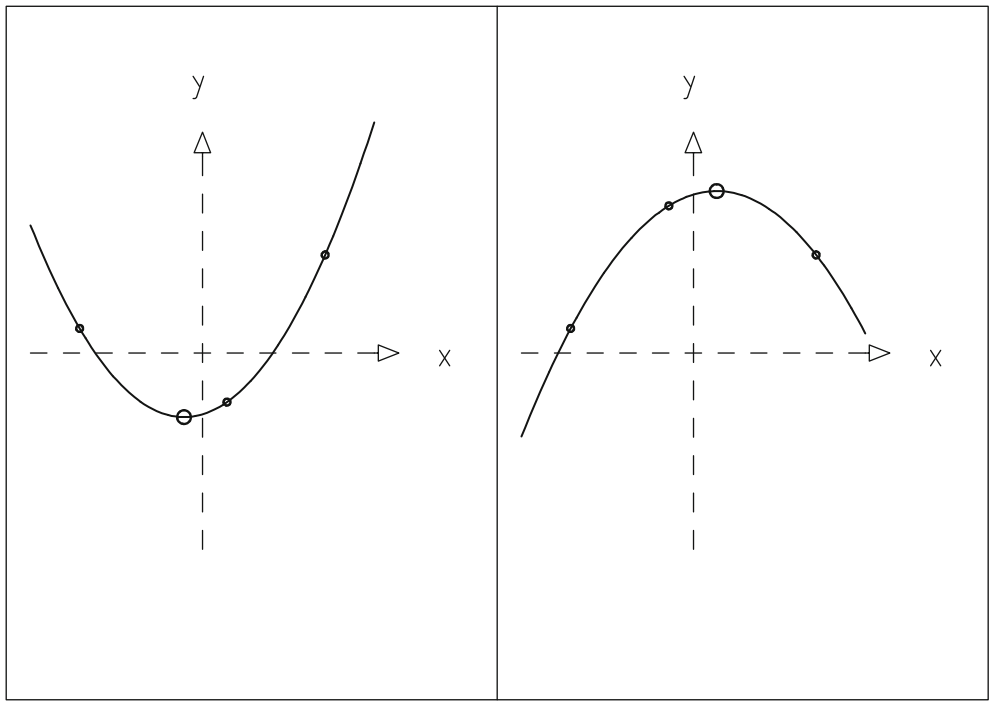

Fig. 10.2: Parabola through three points (*small circles*) and its extremum (*large circle*). In the *left figure* there is a minimum, and on the *right* a maximum.

The class MinParab performs this simple calculation. We must still determine whether the extremum of the parabola is a minimum or a maximum (cf. Fig. 10.2). One has a minimum if the second derivative of (10.2.2) with respect to $x - x_b$ is positive, i.e., if $c_2 > 0$. We now order the three given points such that

$$x_a < x_b < x_c$$

and find that then

$$1/C = (x_c - x_b)(x_a - x_b)(x_c - x_a) = -(x_c - x_b)(x_b - x_a)(x_c - x_a) < 0 \quad .$$

With this one has for the sign of c_2

$$\operatorname{sign} c_2 = \operatorname{sign}[(x_c - x_b)(y_a - y_b) + (x_b - x_a)(y_c - y_b)] \quad .$$

Both expressions $(x_c - x_b)$ and $(x_b - x_a)$ are positive. Therefore for the extremum to be a minimum it is sufficient that

$$y_a > y_b \quad , \qquad y_c > y_b \quad . \tag{10.2.6}$$

The condition is not necessary, but has the advantage of greater clarity. In the interval $x_a < x < x_c$ one clearly has a minimum if there is a point (x_b, y_b) in the interval where the function has a value smaller than at the two end points. Clearly this statement is also valid if the function is not a parabola. We will make use of this fact in the next section.

10.3 Function of *n* Variables on a Line in an *n*-Dimensional Space

Locating the minimum of the function $M(x)$ of a single variable x in an interval of the x axis is equivalent to locating the minimum of a function $M(\mathbf{x})$ of an n-dimensional vector of variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ with respect to a given line in the n-dimensional space. If $\mathbf{x}_0$ is a fixed point and $\mathbf{d}$ is a fixed vector, then

$$\mathbf{x}_0 + a\mathbf{d} \quad , \qquad -\infty < a < \infty \quad , \tag{10.3.1}$$

describes a fixed line (see Fig. 10.3), and

$$f(a) = M(\mathbf{x}_0 + a\mathbf{d}) \tag{10.3.2}$$

is the value of the function at the point a on this line. For $n = 1$, $\mathbf{x}_0 = 0$, $\mathbf{d} = 1$ and with the change of notation $a = x$, that is, $f(x) = M(x)$, one recovers the original problem.

The class FunctionOnline computes the value (10.3.2); it makes use of an extension of the abstract class DatanUserFunction, to be provided by the user, which defines the function $M(\mathbf{x})$. In Sects. 10.4 through 10.6 we consider the minimum of a function of a single variable. The programs also treat, however, the case of a minimum of a function of n variables on a line in the n-dimensional space.

10.4 Bracketing the Minimum

For many minimization procedures it is important to know ahead of time that the minimum x_m is located in a specific interval,

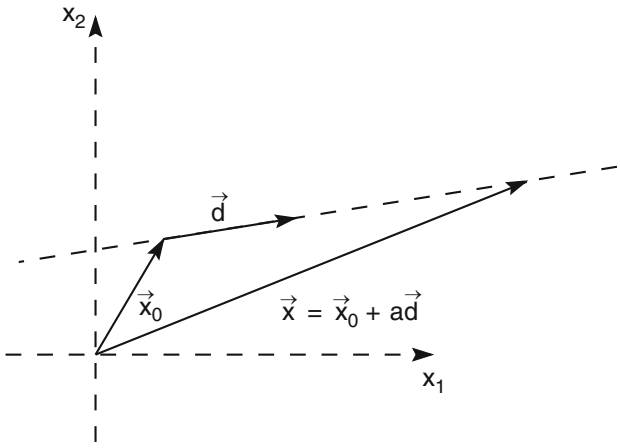

Fig. 10.3: The line given by (10.3.1) in two dimensions.

$$x_a < x_\mathrm{m} < x_c \quad . \tag{10.4.1}$$

By systematically reducing the interval, the position of the minimum can then be further constrained until finally for a given precision ε one has

$$|x_a - x_c| < \varepsilon \quad . \tag{10.4.2}$$

There is, in fact, a minimum in the interval (10.4.1) if there is an x value x_b such that

$$M(x_b) < M(x_a) \quad , \qquad M(x_b) < M(x_c) \quad , \qquad x_a < x_b < x_c \quad . \tag{10.4.3}$$

The class MinEnclose attempts to bracket the minimum of a function by giving values x_a, x_b, x_c with the property (10.4.3). The program is based on a similar subroutine by PRESS et al. [12]. Starting from the input values x_a, x_b, which, if necessary, are relabeled so that $y_b \leq y_a$, a value $x_c = x_b + p(x_b - x_a)$ is computed that presumably lies in the direction of decreasing function value, i.e., closer to the minimum. The factor p in our program is set to $p = 1.618034$. In this way the original interval (x_a, x_b) is enlarged by the ratio of the golden section (cf. Sect. 10.5). The goal is reached if $y_c > y_b$. If this is not the case, a parabola is constructed through the three points $(x_a, y_a), (x_b, y_b), (x_c, y_c)$, whose minimum is at x_m.

We now examine the point $(x_\mathrm{m}, y_\mathrm{m})$. Here one must distinguish between various cases:

(a) $x_b < x_\mathrm{m} < x_c$:

- (a1) $y_\mathrm{m} < y_c$: (x_b, x_m, x_c) is the desired interval.
- (a2) $y_b < y_\mathrm{m}$: (x_a, x_b, x_m) is the desired interval.
- (a3) $y_\mathrm{m} > y_c$ and $y_\mathrm{m} < y_b$: There is no minimum. The interval will be extended further to the right.

(b) $x_c < x_m < x_{end}$ and $x_{end} = x_b + f(x_c - x_b)$ and $f = 10$ in our program.

(b1) $y_m > y_c$: (x_b, x_c, x_m) is the desired interval.

(b2) $y_m < y_c$: There is no minimum. The interval will be extended further to the right.

(c) $x_{end} < x_m$: As a new interval (x_b, x_c, x_{end}) is used.

(d) $x_m < x_b$: This result is actually impossible. It can, however, be caused by a rounding error. The interval will be extended further to the right.

If the goal is not reached in the current step, a further step is taken with the new interval. Figure 10.4 shows an example of the individual steps carried out until the bracketing of the minimum is reached.

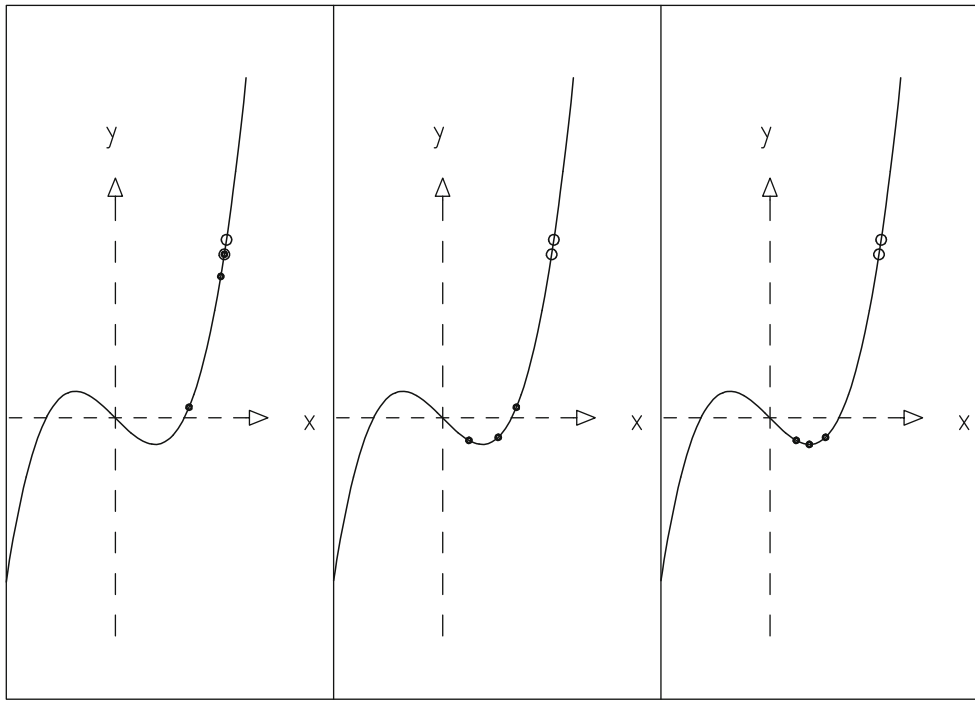

Fig. 10.4: Bracketing of a minimum with three points a, b, c according to (10.4.3). The initial values are shown as *larger circles*, and the results of the individual steps are shown as *small circles*.

10.5 Minimum Search with the Golden Section

As soon as the minimum has been enclosed by giving three points x_a, x_b, x_c with the property (10.4.3), the bracketing can easily be tightened further. One chooses a point x inside the larger of the two subintervals (x_a, x_b) and (x_b, x_c).

If the value of the function at x is smaller than at x_b, then the subinterval containing x is taken as the new interval. If the value of the function is greater, then x is taken as the endpoint of the new interval.

A particularly clever division of the intervals is possible with the *golden section*. Let us assume (see Fig. 10.5) that

$$g = \frac{\ell}{L} \quad , \qquad g > \frac{1}{2} \quad , \tag{10.5.1}$$

is the length of the subinterval (x_a, x_b) (to be determined later) measured in units of the length of the full interval (x_a, x_c). We now want to be able to divide the subinterval (x_a, x_b) again with a point x corresponding to a fraction g,

$$g = \frac{\lambda}{\ell} \quad . \tag{10.5.2}$$

In addition, the points x and x_b should be situated symmetrically with respect to each other in the interval (x_a, x_c), i.e.,

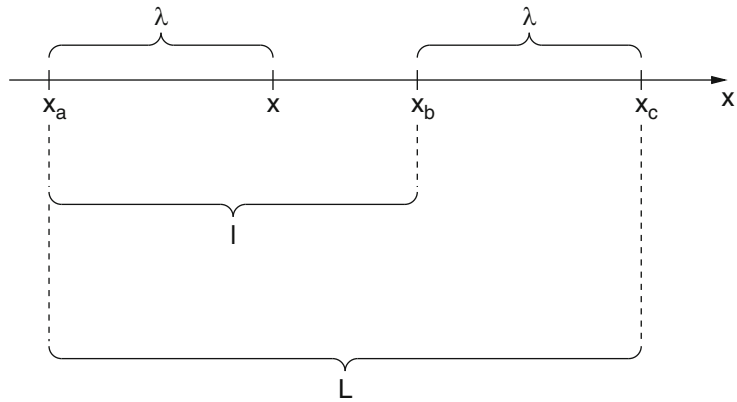

Fig. 10.5: The golden section.

$$\lambda = L - \ell \quad . \tag{10.5.3}$$

It follows that

$$\frac{\ell}{\lambda + \ell} = \frac{\lambda}{\ell} \quad ,$$

that is,

$$\lambda = \frac{\sqrt{5}-1}{2}\ell$$

and

$$g = \frac{\sqrt{5}-1}{2} \approx 0.618\,034 \quad . \tag{10.5.4}$$

As shown at the beginning of this section (for the case shown in Fig. 10.5 $x_b - x_a > x_c - x_b$), the minimum, which was originally only constrained to be in the interval (x_a, x_c), now lies either in the interval (x_a, x_b) or in the interval (x, x_c). By subdividing by the golden section one obtains intervals of equal size.

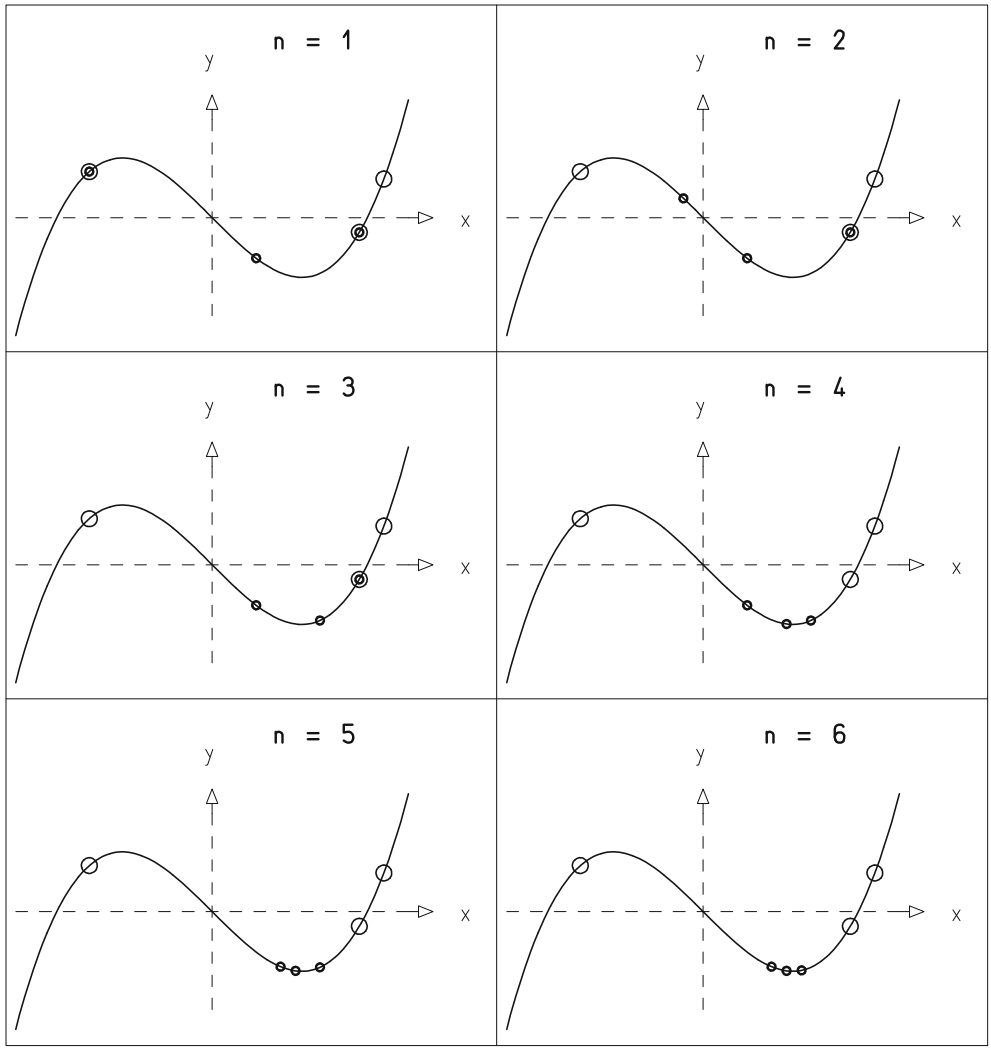

Fig. 10.6: Stepwise bracketing of a minimum by subdividing according to the golden section. The original interval (*larger circles*) is reduced with every step (*small circles*).

Figure 10.6 shows the first six steps of an example of minimization with the golden section.

10.6 Minimum Search with Quadratic Interpolation

From the example shown in Fig. 10.6 one sees that the procedure of interval subdivision is quite certain, but works slowly. We now combine it, therefore, with quadratic interpolation.

The class MinCombined is based on a program developed by BRENT [13], who first combined the two methods. The meanings of the most important variable names in the program are as follows. `a` and `b` denote the x values x_a and x_b that contain the minimum. `xm` is their mean value. `m` is the point at which the function has its lowest value up to this step, `w` is the point with the second lowest, and `v` is the point with the third lowest value of the function. `u` is the point where the function was last computed. One begins with the two initial values, x_a and x_b, that contain the minimum and adds a point x, that divides the interval (x_a, x_b) according to the golden section. Then in each subsequent iteration parabolic interpolation is attempted as in Sect. 10.2. The result is accepted if it lies in the interval defined by the last step *and* if in this step the change in the minimum is less than half as much as in the previous one. By this condition it is ensured that the procedure converges, i.e., that the steps become smaller on the average, although a temporary increase in step size is tolerated. If both conditions are not fulfilled, a reduction of the interval is carried out according to the golden section.

Numerical questions are handled in a particularly careful way of Brent. Starting from the two parameters ε and t, which define the relative precision, and the current value x for the position of the minimum, an absolute precision $\Delta x = \varepsilon x + t = \text{tol}$ is computed according to (10.1.21). The iterations are continued until the half of the interval width falls below the value tol (i.e., until the distance from `x` to `xm` is not greater than tol) or until the maximum number of steps is reached. In addition the function is not computed at points that are separated by a distance less than tol, since such function values would not differ significantly.

The first six steps of an example of minimization according to Brent are shown in Fig. 10.7. Steps according to the golden section are marked by *GS*, and those from quadratic interpolation with *QI*. The comparison with Fig. 10.6 shows the considerably faster convergence achieved by quadratic interpolation.

10.7 Minimization Along a Direction in n Dimensions

The class MinDir computes the minimum of a function of of n variables along the line defined in Sect. 10.3 by $\mathbf{x}_0$ and $\mathbf{d}$. It first uses MinEnclose to bracket the minimum and then MinCombined to locate it exactly.

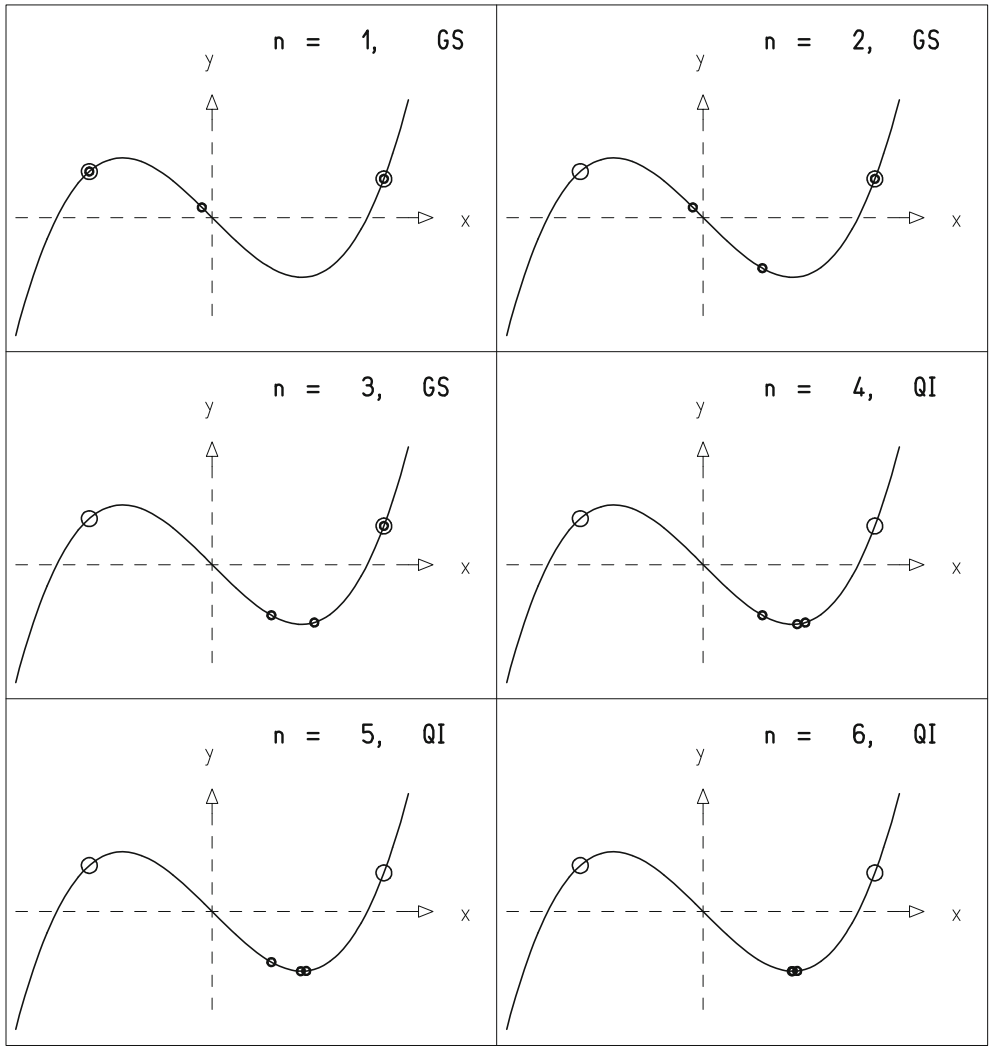

Fig. 10.7: Stepwise bracketing of a minimum with the combined method of Brent. The initial interval (*large circles*) is reduced with each step (*small circles*). Steps are carried out according to the golden section (GS) or by quadratic interpolation (QI).

The class MinDir is the essential tool used to realize a number of different strategies to find a minimum in n dimensional space, which we discuss in Sects. 10.9–10.15. A different type of strategy is the basis of the simplex method presented in the next section.

10.8 Simplex Minimization in n Dimensions

A simple and very elegant (although relatively slow) procedure for determining the minimum of a function of several variables is the simplex method by

NELDER and MEAD [14]. The variables $x_1, x_2, \ldots, x_n$ define an n-dimensional space. A *simplex* is defined in this space by $n+1$ points $\mathbf{x}_i$,

$$\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{ni}) \quad . \tag{10.8.1}$$

A simplex in two dimensions is a triangle with the corner points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

We use y_i to designate the value of the function at $\mathbf{x}_i$ and use particular indices for labeling special points $\mathbf{x}_i$. At the point $\mathbf{x}_H$ the value of the function is highest, i.e., $y_H > y_i, i \neq H$. At the point $\mathbf{x}_h$ it is higher than at all other points except $\mathbf{x}_H$ ($y_h > y_i$, $i \neq H$, $i \neq h$) and at the point $\mathbf{x}_\ell$ it is lowest ($y_\ell < y_i, i \neq \ell$). The simplex is now changed step by step. In each substep one of four operations takes place, these being *reflection*, *stretching*, *flattening*, or *contraction* of the simplex (cf. Fig. 10.8).

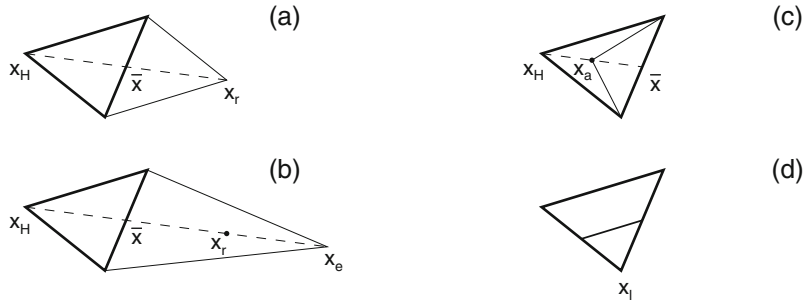

Fig. 10.8: Transformation of a simplex (*triangle* with *thick border*) into a new form (*triangle* with *thin border*) by means of a reflection (**a**), stretching (**b**), flattening (**c**), and contraction (**d**).

Using $\overline{\mathbf{x}}$ to designate the center-of-gravity of the (hyper)surface of the simplex opposite to $\mathbf{x}_H$,

$$\overline{\mathbf{x}} = \frac{1}{N-1} \sum_{i \neq H} \mathbf{x}_i \quad , \tag{10.8.2}$$

the *reflection* of $\mathbf{x}_H$ is

$$\mathbf{x}_r = (1+\alpha)\overline{\mathbf{x}} - \alpha \mathbf{x}_H \tag{10.8.3}$$

with $\alpha > 0$ as the *coefficient of reflection*. The reflected simplex differs from the original one only in that $\mathbf{x}_H$ is replaced by $\mathbf{x}_r$.

A *stretching* of the simplex consists of replacing $\mathbf{x}_H$ by

$$\mathbf{x}_e = \gamma \mathbf{x}_r + (1-\gamma)\overline{\mathbf{x}} \tag{10.8.4}$$

with the *coefficient of stretching* γ. One therefore chooses (for $\gamma > 1$) a point along the line joining $\mathbf{x}_H$ and $\mathbf{x}_r$, which is still further than the point $\mathbf{x}_r$.

In a *flattening*, $\mathbf{x}_H$ is replaced by a point lying on the line joining $\mathbf{x}_H$ and $\overline{\mathbf{x}}$. This point is located between the two points,

$$\mathbf{x}_a = \beta \mathbf{x}_H + (1-\beta)\overline{\mathbf{x}} \quad . \tag{10.8.5}$$

The *coefficient of flattening* β is in the range $0 < \beta < 1$.

In the three operations discussed up to now only one point of the simplex is changed, that being the point $\mathbf{x}_H$, which corresponds to the highest value of the function. The point is displaced along the line by $\mathbf{x}_H$ and $\overline{\mathbf{x}}$. After the displacement it either lies on the same side of $\overline{\mathbf{x}}$ (flattening) or on the other side of $\overline{\mathbf{x}}$ (reflection) or even far beyond $\overline{\mathbf{x}}$ (stretching). In contrast to these operations, in a *contraction*, all points but one are replaced. The point $\mathbf{x}_\ell$ where the function has its lowest value is retained. All remaining points are moved to the midpoints of the line segments joining them with $\mathbf{x}_\ell$

$$\mathbf{x}_{ci} = (\mathbf{x}_i + \mathbf{x}_\ell)/2 \quad , \qquad i \neq \ell \quad . \tag{10.8.6}$$

For the original simplex and for each one created by an operation, the points $\overline{\mathbf{x}}$ and $\mathbf{x}_r$ and the corresponding function values $\overline{y}$ and y_r are computed. The next operation is determined as follows:

(a) If $y_r < y_\ell$, a stretching is attempted. If this gives $y_e < y_\ell$, then the stretching is carried out. Otherwise one performs a *reflection*.

(b) For $y_r > y_h$ a reflection is carried out if $y_r < y_H$. Otherwise the simplex is unchanged. In each case this is followed by a *flattening*. If one obtains as a result of the flattening a point $\mathbf{x}_a$ for which the function value is not less than both y_H and $\overline{y}$, the flattening is rejected, and instead a contraction is carried out.

After every step we examine the quantity

$$r = \frac{|\, y_H - y_\ell \,|}{|\, y_H \,| + |\, y_\ell \,|} \quad . \tag{10.8.7}$$

If this falls below a given value, the procedure is terminated and we regard $\mathbf{x}_\ell$ as the point where the function has its minimum.

The class MinSim determines the minimum of a function of n variables by the simplex method. The program is illustrated in the example of Fig. 10.9. The triangle with the dark border is the initial simplex. The sequence of triangles with thin borders starting from it correspond to the individual transformations. One clearly recognizes as the first steps: stretching, stretching, reflection, reflection, flattening …. The simplex first finds its way into the

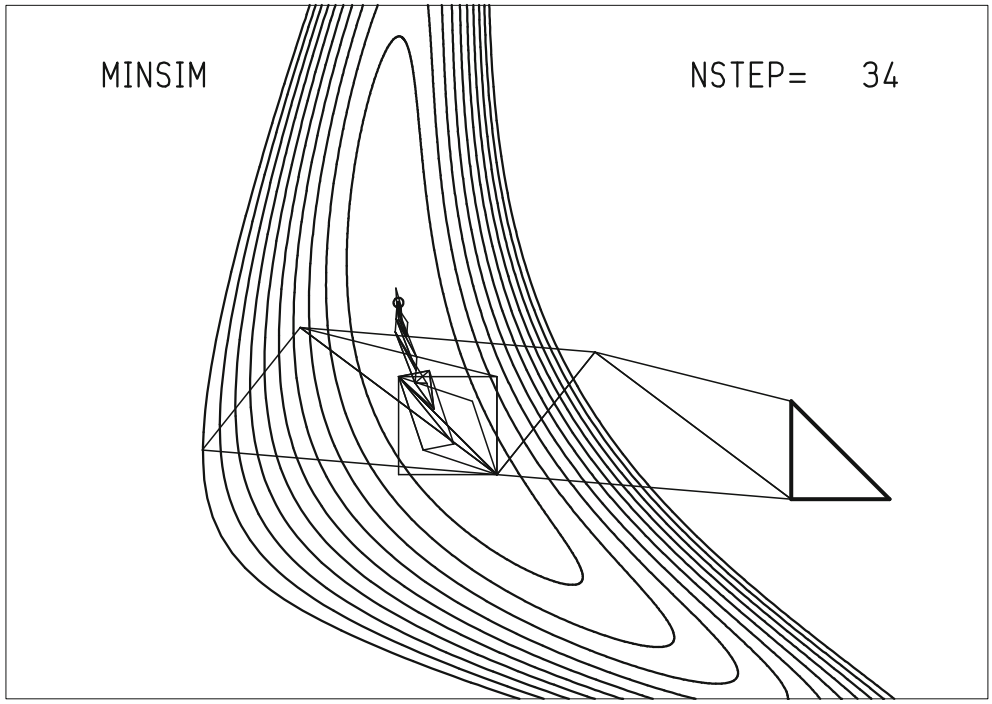

Fig. 10.9: Determining the minimum of a function of two variables with the simplex method. The function is shown by *contour lines* on which the function is constant. The function is highest on the outermost contour. The minimum is at the position of the *small circle* within the innermost contour. Each simplex is a *triangle*. The initial simplex is marked by *thicker lines*.

"valley" of the function and then runs along the bottom of the valley towards the minimum. It is reshaped in the process so that it is longest in the direction in which it is progressing, and in this way it can also pass through narrow valleys. It is worth remarking that this method, which is obtained by considering the problem in two and three dimensions, also works in n dimensions and also in this case possesses a certain descriptiveness.

10.9 Minimization Along the Coordinate Directions

Some of the methods for searching for a minimum in an n-dimensional space are based on the following principle. Starting from a point $\mathbf{x}_0$ one searches for the minimum along a given direction in the space. Next one minimizes from there along another direction and finds a new minimum, etc. Within this general framework various *strategies* can be developed to choose the individual directions.

The simplest strategy consists in the choice of the coordinate directions in the space of the n variables x_i. We can label the corresponding basis vectors with $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$, and they are then chosen in order as directions. After $\mathbf{e}_n$ one begins again with $\mathbf{e}_1, \mathbf{e}_2, \ldots$. A partial sequence of minimizations along all coordinate directions starting from $\mathbf{x}_0$ gives the point $\mathbf{x}_1$. After a new partial sequence one obtains $\mathbf{x}_2$, etc.

The procedure is ended successfully when for the values of the function $M(\mathbf{x}_n)$ and $M(\mathbf{x}_{n-1})$ for two successive steps one has

$$M(\mathbf{x}_{n-1}) - M(\mathbf{x}_n) < \varepsilon |M(\mathbf{x}_n)| + t \quad , \tag{10.9.1}$$

i.e., a condition corresponding to (10.1.21) for given values of ε and t. We compare here, however, the value of the function M and not the independent variables $\mathbf{x}$. Otherwise we would have to compute the distance between two points in an n-dimensional space.

Figure 10.10 shows the minimization of the same function as in Fig. 10.9 with the coordinate-direction method. After the first comparatively large step, which leads into the "valley" of the function, the following steps are quite small. The individual directions are naturally perpendicular to each other. The "best" point at each step moves along a staircase-like path along the floor of the valley towards the minimum.

10.10 Conjugate Directions

The slow convergence seen in Fig. 10.10 clearly stems from the fact that when minimizing along one direction, one loses the result of the minimization with respect to another direction. We will now try to choose the directions in such a way that this does not happen. For this we suppose for simplicity that the function is a quadratic form (10.1.10). Its gradient at the point $\mathbf{x}$ is then given by (10.1.12),

$$\nabla M = -\mathbf{b} + A\mathbf{x} \quad . \tag{10.10.1}$$

The change of the gradient in moving by $\Delta\mathbf{x}$ is

$$\Delta(\nabla M) = \nabla M(\mathbf{x} + \Delta\mathbf{x}) - \nabla M(\mathbf{x}) = A\mathbf{x} + A\Delta\mathbf{x} - A\mathbf{x} = A\Delta\mathbf{x} \quad . \tag{10.10.2}$$

This expression is a vector that gives the direction of change of the gradient when the argument is moved in the direction $\Delta\mathbf{x}$.

If a minimization has been carried out along a direction $\mathbf{p}$, then the reduction in M with respect to $\mathbf{p}$ is retained when one moves in a direction $\mathbf{q}$ such that the gradient only changes perpendicular to $\mathbf{p}$, i.e., when one has

$$\mathbf{p} \cdot (A\mathbf{q}) = \mathbf{p}^{\mathrm{T}} A\mathbf{q} = 0 \quad . \tag{10.10.3}$$

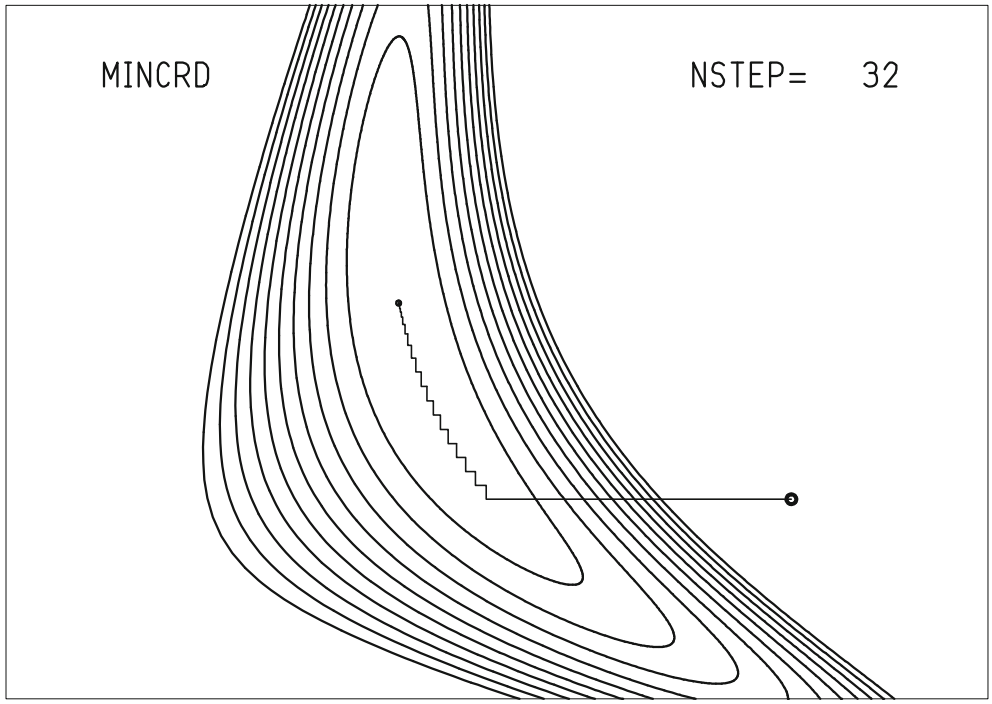

Fig. 10.10: Minimization along the coordinate directions. The starting point is shown by the *larger circle*, and the end by the *smaller circle*. The *line* shows the results of the individual steps.

The vectors $\mathbf{p}$ and $\mathbf{q}$ are said to be *conjugate* to each other with respect to the positive-definite matrix A. If one has n variables, i.e., if A is an $n \times n$ matrix, then one can find in general n conjugate linearly independent vectors.

POWELL [15] has given a method to find a set of conjugate directions for a function described by a quadratic form. For this one chooses as a first set of directions n linearly independent unit vectors $\mathbf{p}_i$, e.g., the coordinate directions $\mathbf{p}_i = \mathbf{e}_i$. Starting from a point $\mathbf{x}_0$ one finds successively the minima in the directions $\mathbf{p}_i$. The results can be labeled by

$$\begin{aligned}
\mathbf{a}_1 &= \alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \cdots + \alpha_n\mathbf{p}_n \quad , \\
\mathbf{a}_2 &= \alpha_2\mathbf{p}_2 + \cdots + \alpha_n\mathbf{p}_n \quad , \\
&\vdots \\
\mathbf{a}_n &= \alpha_n\mathbf{p}_n \quad .
\end{aligned}$$

Here $\mathbf{a}_1$ is the vector representing all n substeps, $\mathbf{a}_2$ contains all of the steps except the first one, and $\mathbf{a}_n$ is just the last substep. The sum of the n substeps leads then from the point $\mathbf{x}_0$ to

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{a}_1 \, .$$

The direction $\mathbf{a}_1$ describes the average direction of the first n substeps. Therefore we carry out a step in the direction $\mathbf{a}_1$, call the result again $\mathbf{x}_0$, determine a new set of directions

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{p}_2 \quad , \\ \mathbf{q}_2 &= \mathbf{p}_3 \quad , \\ &\vdots \\ \mathbf{q}_{n-1} &= \mathbf{p}_n \quad , \\ \mathbf{q}_n &= \mathbf{a}_1/|\mathbf{a}_1| \quad , \end{aligned}$$

then call these $\mathbf{q}_i$ again $\mathbf{p}_i$ and proceed as above. As was shown by POWELL [15], the directions after n steps, i.e., $n(n+1)$ individual minimizations, are individually conjugate to each other if the function is a quadratic form.

10.11 Minimization Along Chosen Directions

The procedure given at the end of the last section contains, however, the danger that the directions $\mathbf{p}_1, \ldots, \mathbf{p}_n$ can become almost linearly dependent, because at every step $\mathbf{p}_1$ is rejected in favor of $\mathbf{a}_1/|\mathbf{a}_1|$, and these directions need not be very different from step to step. Powell has therefore suggested not to replace the direction $\mathbf{p}_1$ by $\mathbf{a}_1/|\mathbf{a}_1|$, but rather to replace the direction $\mathbf{p}_{\max}$, along which the greatest reduction of the function took place. This sounds at first paradoxical, since what is clearly the best direction is replaced by another. But since out of all the directions $\mathbf{p}_{\max}$ gave the largest contribution towards reducing the function, $\mathbf{a}_1/|\mathbf{a}_1|$ has a significant component in the direction $\mathbf{p}_{\max}$. By retaining these two similar directions one would increase the danger of a linear dependence.

In some cases, however, after completing a step we will retain the old directions without change. We denote by $\mathbf{x}_0$ the point before carrying out the step in progress, by $\mathbf{x}_1$ the point obtained by this step, by

$$\mathbf{x}_e = \mathbf{x}_1 + (\mathbf{x}_1 - \mathbf{x}_0) = 2\mathbf{x}_1 - \mathbf{x}_0 \tag{10.11.1}$$

an extrapolated point that lies in the new direction $\mathbf{x}_1 - \mathbf{x}_0$ from $\mathbf{x}_0$ but is still further than $\mathbf{x}_1$, and by M_0, M_1, and M_e the corresponding function values. If

$$M_e \geq M_0 \quad , \tag{10.11.2}$$

then the function no longer decreases significantly in the direction $(\mathbf{x}_0 - \mathbf{x}_1)$. We then remain with the previous directions.

Further we denote by ΔM the greatest change in M along a direction in the step in progress and compute the quantity

$$T = 2(M_0 - 2M_1 + M_e)(M_0 - M_1 - \Delta M)^2 - (M_0 - M_e)^2 \Delta M \quad . \quad (10.11.3)$$

If

$$T \geq 0 \quad ,$$

then we retain the old directions. This requirement is satisfied if either the first or second factor in the first term of (10.11.3) becomes large. The first factor $(M_0 - 2M_1 + M_e)$ is proportional to a second derivative of the function M. If it is large (compared to the first derivative in meaningful units), then we are already close to the minimum. The second factor $(M_0 - M_1 - \Delta M)^2$ is large when the reduction of the function $M_0 - M_1$, with the contribution ΔM, does not stem mainly from a single direction.

The class MinPow determines the minimum of a function of n variables by successive minimization along chosen directions according to Powell. Figure 10.11 shows the advantages of that method. One can see a significantly faster convergence to the minimum compared to that of Fig. 10.10.

10.12 Minimization in the Direction of Steepest Descent

In order to get from a point $\mathbf{x}_0$ to the minimum of a function $M(\mathbf{x})$, it is sufficient to always follow the negative gradient $\mathbf{b}(\mathbf{x}) = -\nabla M(\mathbf{x})$. It is thus obvious that one should look for the minimum along the direction $\nabla M(\mathbf{x}_0)$. One calls this point $\mathbf{x}_1$, searches then for the minimum along $\nabla M(\mathbf{x}_1)$, and so forth, until the termination requirement (10.9.1) is satisfied.

A comparison of the steps taken with this method, shown in Fig. 10.12, with those from minimization along a coordinate direction (Fig. 10.10) shows, however, a surprising similarity. In both cases, successive steps are perpendicular to each other. Thus the directions cannot be fitted to the function in the course of the procedure and convergence is slow.

The initially surprising fact that successive gradient directions are perpendicular to each other stems from the construction of the procedure. Searching from $\mathbf{x}_1$ for the minimum in the direction $\mathbf{b}_0$ means that the derivative in the direction $\mathbf{b}_0$ vanishes at the point $\mathbf{x}_1$,

$$\mathbf{b}_0 \cdot \nabla M(\mathbf{x}_1) = 0 \quad ,$$

and thus the gradient is perpendicular to $\mathbf{b}_0$.

10.13 Minimization Along Conjugate Gradient Directions

We take up again the idea of conjugate directions from Sect. 10.10. We construct, starting from an arbitrary vector $\mathbf{g}_1 = \mathbf{h}_1$, two sequences of vectors,

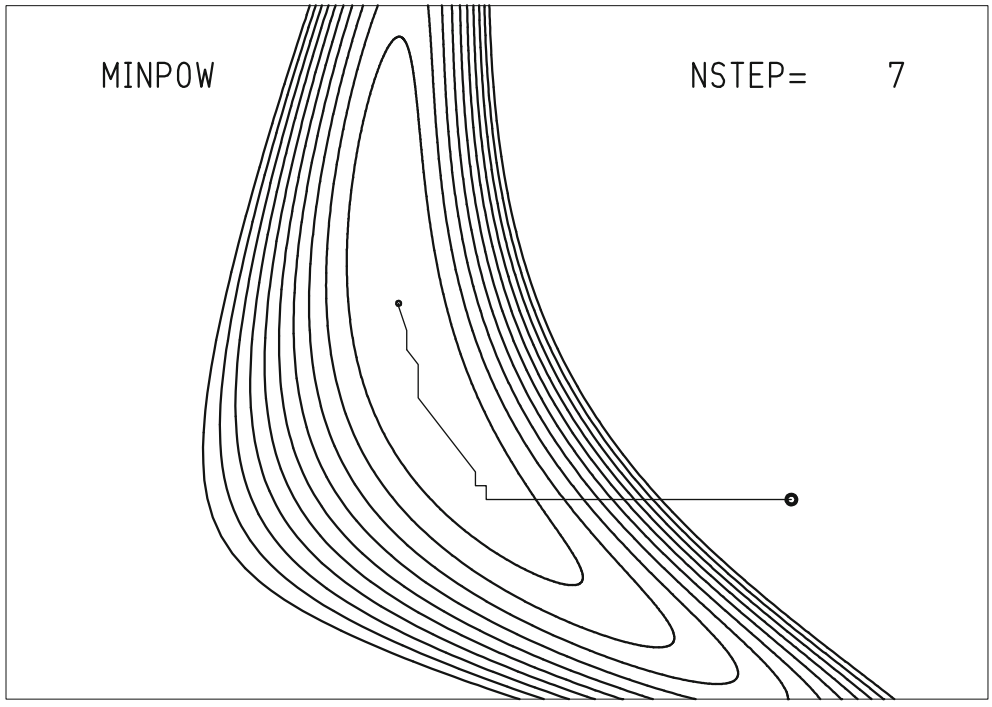

Fig. 10.11: Minimization along a chosen direction with the method of Powell.

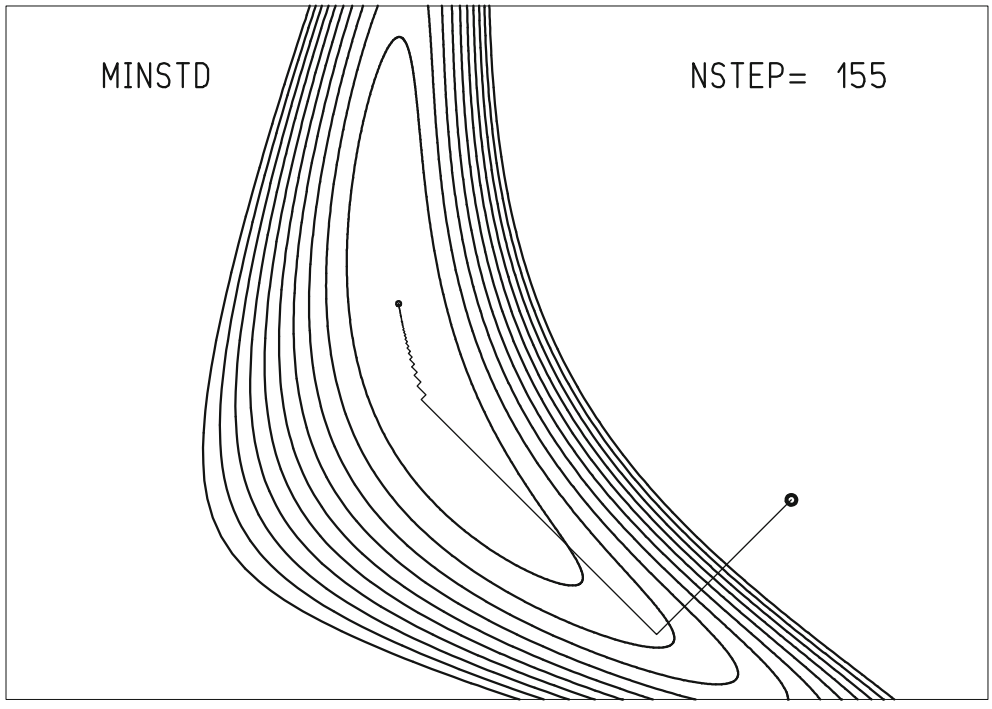

Fig. 10.12: Minimization in the direction of steepest descent.

$$\mathbf{g}_{i+1} = \mathbf{g}_i - \lambda_i A\mathbf{h}_i \quad , \qquad i = 1, 2, \ldots \quad , \tag{10.13.1}$$

$$\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i \quad , \qquad i = 1, 2, \ldots \quad , \tag{10.13.2}$$

with

$$\lambda_i = \frac{\mathbf{g}_i^{\mathrm{T}} \mathbf{g}_i}{\mathbf{g}_i^{\mathrm{T}} A \mathbf{h}_i} \quad , \qquad \gamma_i = \frac{\mathbf{g}_{i+1}^{\mathrm{T}} A \mathbf{h}_i}{\mathbf{h}_i^{\mathrm{T}} A \mathbf{h}_i} \quad . \tag{10.13.3}$$

Thus one has

$$\mathbf{g}_{i+1}^{\mathrm{T}} \mathbf{g}_i = 0 \quad , \tag{10.13.4}$$

$$\mathbf{h}_{i+1}^{\mathrm{T}} A \mathbf{h}_i = 0 \quad . \tag{10.13.5}$$

This means that successive vectors $\mathbf{g}$ are orthogonal, and successive vectors $\mathbf{h}$ are conjugate to each other.

Rewriting the relation (10.13.3) gives

$$\gamma_i = \frac{\mathbf{g}_{i+1}^{\mathrm{T}} \mathbf{g}_{i+1}}{\mathbf{g}_i^{\mathrm{T}} \mathbf{g}_i} = \frac{(\mathbf{g}_{i+1} - \mathbf{g}_i)^{\mathrm{T}} \mathbf{g}_{i+1}}{\mathbf{g}_i^{\mathrm{T}} \mathbf{g}_i} \quad , \tag{10.13.6}$$

$$\lambda_i = \frac{\mathbf{g}_i^{\mathrm{T}} \mathbf{h}_i}{\mathbf{h}_i^{\mathrm{T}} A \mathbf{h}_i} \quad . \tag{10.13.7}$$

We now try to construct the vectors $\mathbf{g}_i$ and $\mathbf{h}_i$ without explicit knowledge of the Hessian matrix A. To do this we again assume that the function to be minimized is a quadratic form (10.1.10). At a point $\mathbf{x}_i$ we define the vector $\mathbf{g}_i = -\nabla M(\mathbf{x}_i)$. If we now search for the minimum starting from $\mathbf{x}_i$ along the direction $\mathbf{h}_i$, we find it at $\mathbf{x}_{i+1}$ and construct there $\mathbf{g}_{i+1} = -\nabla M(\mathbf{x}_{i+1})$. Then $\mathbf{g}_{i+1}$ and $\mathbf{g}_i$ are orthogonal, since from (10.1.12) one has

$$\mathbf{g}_i = -\nabla M(\mathbf{x}_i) = \mathbf{b} - A\mathbf{x}_i$$

and

$$\mathbf{g}_{i+1} = -\nabla M(\mathbf{x}_{i+1}) = b - A(\mathbf{x}_i + \lambda_i \mathbf{h}_i) = \mathbf{g}_i - \lambda_i A\mathbf{h}_i \quad . \tag{10.13.8}$$

Here λ_i has been chosen such that $\mathbf{x}_{i+1}$ is the minimum along the direction $\mathbf{h}_i$. This means that there the gradient is perpendicular to $\mathbf{h}_i$,

$$\mathbf{h}_i^{\mathrm{T}} \nabla M(\mathbf{x}_{i+1}) = -\mathbf{h}_i^{\mathrm{T}} \mathbf{g}_{i+1} = 0 \quad . \tag{10.13.9}$$

Substituting into (10.13.8) gives indeed

$$0 = \mathbf{h}_i^{\mathrm{T}} \mathbf{g}_{i+1} = \mathbf{h}_i^{\mathrm{T}} \mathbf{g}_i - \lambda_i \mathbf{h}_i^{\mathrm{T}} A \mathbf{h}_i \quad ,$$

in agreement with (10.13.7).

From these results we can now construct the following algorithm. We begin at $\mathbf{x}_0$, construct there the gradient $\nabla M(\mathbf{x}_0)$, and set its negative equal to the two vectors

$$\mathbf{g}_1 = -\nabla M(\mathbf{x}_0) \quad , \qquad \mathbf{h}_1 = -\nabla M(\mathbf{x}_0) \quad .$$

We minimize along $\mathbf{h}_1$. At the point of the minimum $\mathbf{x}_1$ we construct $\mathbf{g}_2 = -\nabla M(\mathbf{x}_1)$ and compute from (10.13.6)

$$\gamma_1 = \frac{(\mathbf{g}_2 - \mathbf{g}_1)^{\mathrm{T}} \mathbf{g}_1}{\mathbf{g}_1^{\mathrm{T}} \mathbf{g}_1}$$

and from (10.13.2)

$$\mathbf{h}_2 = \mathbf{g}_1 + \gamma_1 \mathbf{h}_1 \quad .$$

From $\mathbf{x}_1$ we then minimize along $\mathbf{h}_2$, and so forth.

The class `MinCjg` determines the minimum of a function of n variables by successive minimization along conjugate gradient directions. A comparison of Fig. 10.13 with Fig. 10.12 shows the superiority of the method of conjugate gradients over determination of direction according to steepest descent, especially near the minimum.

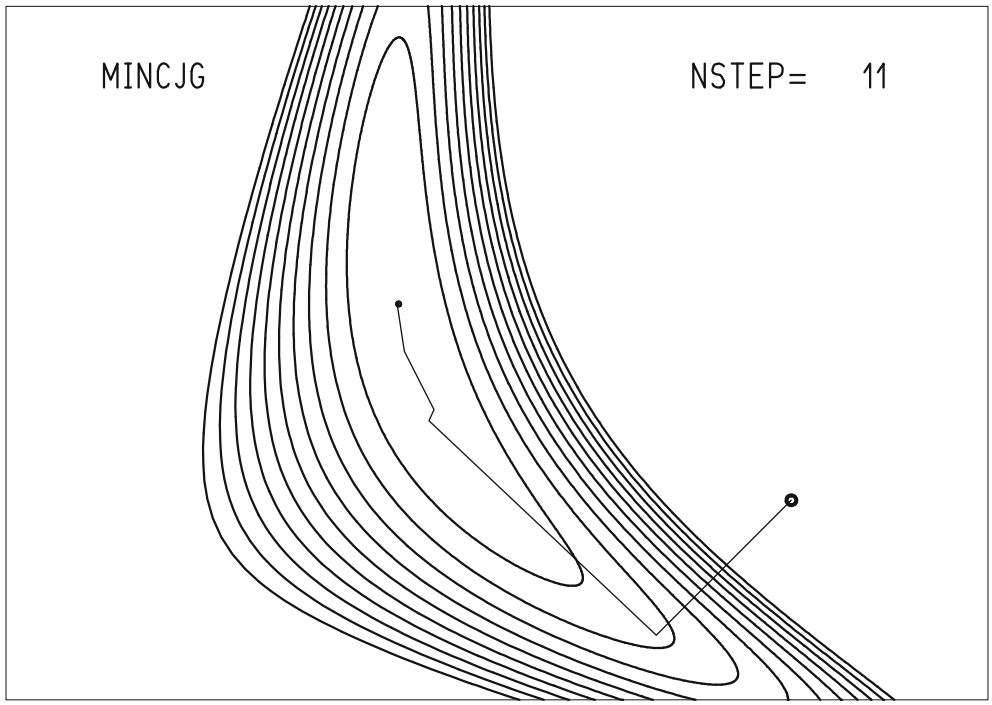

Fig. 10.13: Minimization along conjugate gradient directions.

10.14 Minimization with the Quadratic Form

If the function to be minimized $M(\mathbf{x})$ is of the simple form (10.1.10), then the position of the minimum is given directly by (10.1.13). Otherwise one can always expand $M(\mathbf{x})$ about a point $\mathbf{x}_0$,

$$M(\mathbf{x}) = M(\mathbf{x}_0) - b(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}} A(\mathbf{x}-\mathbf{x}_0) + \cdots \tag{10.14.1}$$

with

$$\mathbf{b} = -\nabla M(\mathbf{x}_0) \quad , \qquad A_{ik} = \frac{\partial^2 M}{\partial x_i \partial x_k} \tag{10.14.2}$$

and obtain as an approximation for the minimum

$$\mathbf{x}_1 = \mathbf{x}_0 + A^{-1}\mathbf{b} \quad . \tag{10.14.3}$$

One can now compute again $\mathbf{b}$ and A as derivatives at the point $\mathbf{x}_1$ and from this obtain a further approximation $\mathbf{x}_2$ according to (10.14.3), and so forth.

For the case where the approximation (10.14.2) gives a good description of the function $M(\mathbf{x})$, the procedure converges quickly, since it tries to jump directly to the minimum. Otherwise it might not converge at all. We have already discussed the difficulties for the corresponding one-dimensional case in Sect. 10.1 with Fig. 10.1.

The class `MinQdr` finds the minimum of a function of n variables with the quadratic form. Figure 10.14 illustrates the operation of the method. One can observe that the minimum is in fact reached in very few steps.

10.15 Marquardt Minimization

MARQUARDT [16] has given a procedure that combines the speed of minimization with the quadratic form near the minimum with the robustness of the method of steepest descent, where one finds the minimum even starting from a point far away. It is based on the following simple consideration.

The prescription (10.14.3), written as a computation of the ith approximation for the position of the minimum,

$$\mathbf{x}_i = \mathbf{x}_{i-1} + A^{-1}\mathbf{b} \quad , \tag{10.15.1}$$

means that one obtains $\mathbf{x}_i$ from the point $\mathbf{x}_{i-1}$ by taking a step along the vector $A^{-1}\mathbf{b}$. Here $\mathbf{b} = -\nabla M(\mathbf{x}_{i-1})$ is the negative gradient, i.e., a vector in the direction of steepest descent of the function M at the point $\mathbf{x}_{i-1}$. If in (10.15.1) one had the unit matrix multiplied by a constant instead of the matrix A, i.e., if instead of (10.15.1) one were to use the prescription

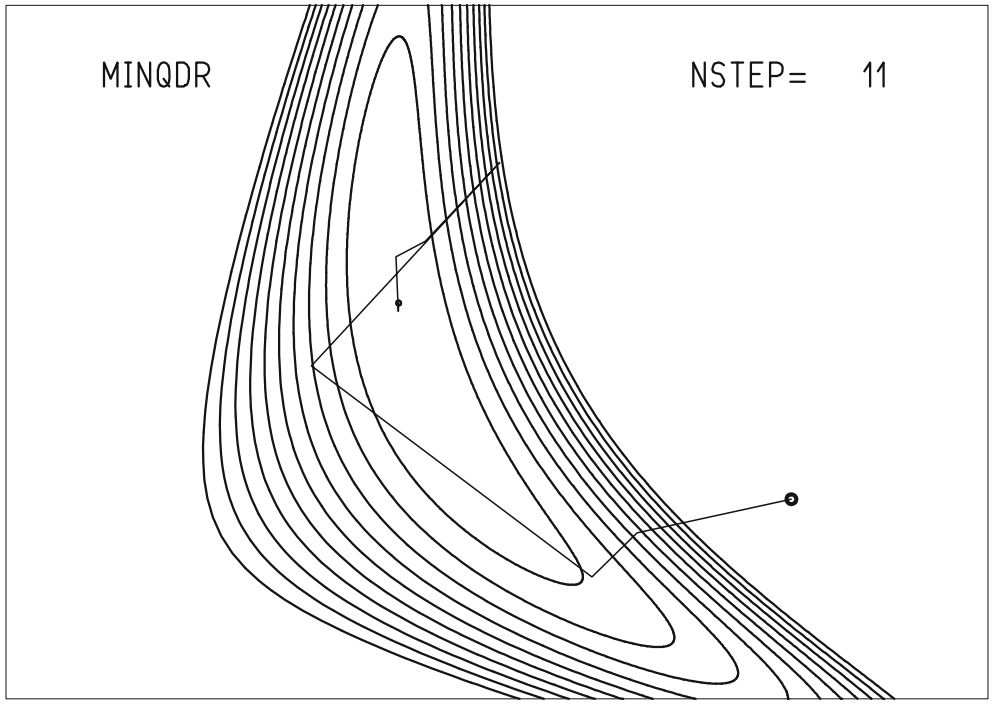

Fig. 10.14: Minimization with quadratic form.

$$\mathbf{x}_i = \mathbf{x}_{i-1} + (\lambda I)^{-1}\mathbf{b} \quad , \tag{10.15.2}$$

then one would step by the vector $\mathbf{b}/\lambda$ from $\mathbf{x}_{i-1}$. This is a step in the direction of steepest descent of the function, which is smaller for larger values of the constant λ. A sufficiently small step in the direction of steepest descent is, however, always a step towards the minimum (at least when one is still in the "approach" to the minimum, i.e., in the one-dimensional case of Fig. 10.1, between the two maxima). The Marquardt procedure consists of interpolating between the prescriptions (10.15.1) and (10.15.2) in such a way that the function M is reduced with every step, and such that the fast convergence of (10.15.1) is exploited as much as possible.

In place of (10.15.1) or (10.15.2) one computes

$$\mathbf{x}_i = \mathbf{x}_{i-1} + (A + \lambda I)^{-1}\mathbf{b} \quad . \tag{10.15.3}$$

Here λ is determined in the following way. One first chooses a fixed number $\nu > 1$ and denotes by $\lambda^{(i-1)}$ the value of λ from the previous step. As an initial value one chooses, e.g., $\lambda^{(0)} = 0.01$. The value obtained from (10.15.3) of $\mathbf{x}_i$ clearly depends on λ. One computes two points $\mathbf{x}_i(\lambda^{(i-1)})$ and $\mathbf{x}_i(\lambda^{(i-1)}/\nu)$, where for λ one chooses the values $\lambda^{(i-1)}$ and $\lambda^{(i-1)}/\nu$, and the corresponding function values $M_i = M(\mathbf{x}_i(\lambda^{(i-1)}))$ and $M_i^{(\nu)} = M(\mathbf{x}_i(\lambda^{(i-1)}/\nu))$. These

are compared with the function value $M_{i-1} = M(\mathbf{x}_{i-1})$. The result of the comparison determines what happens next. The following cases are possible:

(i) $M_i^{(\nu)} \leq M_{i-1}$:
One sets $\mathbf{x}_i = \mathbf{x}_i(\lambda^{(i-1)}/\nu)$ and $\lambda^{(i)} = \lambda^{(i-1)}/\nu$.

(ii) $M_i^{(\nu)} > M_{i-1}$ and $M_i \leq M_{i-1}$:
One sets $\mathbf{x}_i = \mathbf{x}_i(\lambda^{(i-1)})$ and $\lambda^{(i)} = \lambda^{(i-1)}$.

(iii) $M_i^{(\nu)} > M_{i-1}$ and $M_i > M_{i-1}$:
One replaces $\lambda^{(i-1)}$ by $\lambda^{(i-1)}\nu$ and repeats the computation of $\mathbf{x}_i(\lambda^{(i-1)}/\nu)$ and $\mathbf{x}_i(\lambda^{(i-1)})$ and the corresponding function values, and repeats the comparisons.

In this way one ensures that the function value in fact decreases with each step and that the value of λ is always as small as possible when adjusted to the local situation. Clearly (10.15.3) becomes (10.15.1) for $\lambda \to 0$, i.e., it describes minimization with the quadratic form. For very large values of λ, Eq. (10.15.3) becomes the relation (10.15.2), which prescribes a small but sure step in the direction of steepest descent.

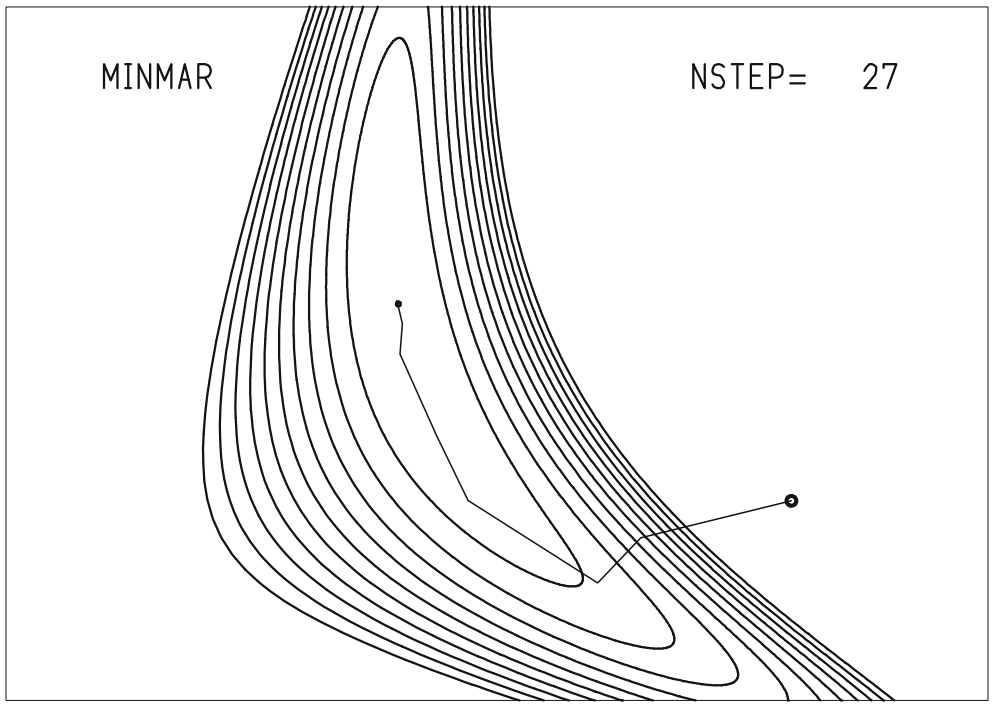

Fig. 10.15: Minimization with the Marquardt procedure.

The class MinMar finds the minimum of a function of n variables by Marquardt minimization. In Fig. 10.15 one sees the operation of the

Marquardt method. It follows a systematic path to the minimum. Comparing with Fig. 10.14 one sees that the method of the quadratic form converged in fewer steps. The Marquardt procedure, however, leads to the minimum in many cases where the method of the quadratic form would fail, e.g., in cases with poorly determined initial values.

10.16 On Choosing a Minimization Method

Given the variety of minimization methods, the user is naturally faced with the question of choosing a method appropriate to the task. Before we give recommendations for this, we will first recall the various methods once again.

The *simplex method* (routine MinSim) is particularly robust. Only function values $M(\mathbf{x})$ are computed. The method is slow, however. Faster, but still quite robust is *minimization along a chosen direction* (routine MinPow). This also requires only function values.

The *method of conjugate gradients* (routine MinCjg) requires, as the name indicates, the computation of not only the function, but also its gradient. The number of iteration steps is, however, approximately equal to that of MinPow.

For *minimization with the quadratic form* (routine MinQdr) and *Marquardt minimization* (routine MinMar) one requires in addition the Hessian matrix of second derivatives. The derivatives are computed numerically with utility routines. The user can replace these utility routines by routines in which the analytic formulas for the derivatives are programmed. If the starting values are sufficiently accurate, MinQdr converges after just a few steps. The convergence is slower for MinMar. The method is, however, more robust; it often converges when starting from values from which MinQdr would fail.

From these characteristics of the methods we arrive at the following recommendations:

1. For problems that need only to be solved once or not many times, that is, in which the computing time does not play an important role, one should choose MinSim or MinPow.

2. For problems occurring repeatedly (with different numerical values) one should use MinMar. If one always has an accurate starting approximation, MinQdr can be used.

3. For repeated problems the derivatives needed for MinMar or MinQdr should be calculated analytically. Although this entails additional programming work, one gains precision compared to numerical derivatives and saves in many cases computing time.

At this point we should make an additional remark about the comparison of the minimization methods of this chapter with the method of least squares from Chap. 9. The method of least squares is a special case of minimization. The function to be minimized is a sum of squares, e.g., (9.1.8), or the generalization of a sum of squares, e.g., (9.5.9). In this generalized sum of squares there appears the matrix of derivatives A. These are not, however, the derivatives of the function to be minimized, but rather derivatives of a function $\mathbf{f}$, which characterizes the problem under consideration; cf. (9.5.2). Second derivatives are never needed. In addition, if one uses the singular value decomposition, as has been done in our programs in Chap. 9 to solve the least-squares problem, one works in numerically critical cases with a considerably higher accuracy than in the computation of sums of squares (cf. Sect. A.13, particularly Example A.4).

Least-squares problems should therefore always be carried out with the routines of Chap. 9. This applies particularly to problems of fitting of functions like those in the examples of Sect. 9.6, when one has many measured points. The matrix A then contains many rows, but few columns. In computing the function to be minimized the product $A^{\mathrm{T}}A$ is encountered, and one is threatened with the above mentioned loss of accuracy in comparison to the singular value decomposition.

10.17 Consideration of Errors

In data analysis minimization procedures are used for determining best estimates $\widetilde{\mathbf{x}}$ for unknown values $\mathbf{x}$. The function to be minimized $M(\mathbf{x})$ is here usually a sum of squares (Chap. 9) or a log-likelihood function (multiplied by -1) as in Chap. 7. In using the equations from Chap. 7 one must note, however, that there the n-vector that was called $\boldsymbol{\lambda}$ is now denoted by $\mathbf{x}$. The variables in Chap. 7 that were called $x^{(i)}$ are the measured values (usually called $\mathbf{y}$ in Chap. 9).

Information on the error of $\widetilde{\mathbf{x}}$ is obtained directly from results of Sects. 9.7, 9.8, and 9.13 by defining

$$H_{ik} = \left(\frac{\partial^2 M}{\partial x_i \partial x_k} \right)_{\mathbf{x}=\widetilde{\mathbf{x}}} \tag{10.17.1}$$

as the elements of the symmetric matrix of second derivatives (the *Hessian matrix*) of the minimization function. Here one must note that the factor f_{QL} takes on the numerical value

$$f_{QL} = 1 \tag{10.17.2}$$

when the function being minimized is a sum of squares. If the function is a log-likelihood function (times -1) then this must be set equal to

$$f_{QL} = 1/2 \quad . \tag{10.17.3}$$

1. Covariance matrix. Symmetric errors. The covariance matrix of $\widetilde{\mathbf{x}}$ is

$$C_{\widetilde{x}} = 2 f_{QL} H^{-1} \quad . \tag{10.17.4}$$

The square roots of the diagonal elements are the (symmetric) *errors*

$$\Delta\widetilde{x}_i = \sqrt{c_{ii}} \quad . \tag{10.17.5}$$

It is only meaningful to give the covariance matrix if the measurement errors are small and/or there are many measurements, i.e., the requirements for (7.5.8) are fulfilled.

2. Confidence ellipsoid. Symmetric confidence limits. The covariance matrix defines the covariance ellipsoid; cf. Sects. 5.10 and A.11. Its center is given by $\mathbf{x} = \widetilde{\mathbf{x}}$. The probability that the true value of $\mathbf{x}$ is contained inside the ellipsoid is given by (5.10.20). The ellipsoid for which this probability has a given value W, the *confidence level*, is given by the *confidence matrix*

$$C_{\widetilde{x}}^{(W)} = \chi_W^2(n_f) C_{\widetilde{x}} \quad . \tag{10.17.6}$$

Here $\chi_W^2(n_f)$ is the quantile of the χ^2-distribution for n_f degrees of freedom and probability $P = W$; cf. (5.10.19) and (C.5.3). The number of degrees of freedom n_f is equal to the number of measured values minus the number of parameters determined by the minimization. The square roots of the diagonal elements of $C_{\widetilde{x}}^{(W)}$ are the distances from the *symmetric confidence limits*,

$$x_{i\pm}^{(W)} = \widetilde{x}_i \pm \sqrt{c_{ii}^{(W)}} \quad . \tag{10.17.7}$$

The class `MinCov` yields the covariance or confidence matrix, respectively, for parameters determined by minimization.

3. Confidence region. If giving a covariance or confidence ellipsoid is not meaningful, it is still possible to give a confidence region with confidence level W. It is determined by the hypersurface

$$M(\mathbf{x}) = M(\widetilde{\mathbf{x}}) + \chi_W^2(n_f) f_{QL} \quad . \tag{10.17.8}$$

With the help of the following routine a contour of a cross section through this hypersurface is drawn in a plane that contains the point $\widetilde{\mathbf{x}}$ and is parallel to (x_i, x_j). Here x_i and x_j are two components of the vector of parameters $\mathbf{x}$. The boundary of the confidence region in a plane spanned by two parameters, which were found by minimization, can be graphically shown with the method `DatanGraphics.drawContour`, cf. Examples 10.1–10.3 and Example Programs 10.2–10.4.

4. Asymmetric errors and confidence limits. If the confidence region is not an ellipsoid, the asymmetric *confidence limits* for the variable x_i can still be determined by

$$\min\left\{M(\widetilde{\mathbf{x}});\, x_i = x_{i\pm}^{(W)}\right\} = M(\widetilde{\mathbf{x}}) + \chi_W^2(n_f)\, f_{QL} \quad . \tag{10.17.9}$$

The differences

$$\begin{aligned} \Delta x_{i+}^{(W)} &= x_{i+}^{(W)} - \widetilde{x}_i \quad , \\ \Delta x_{i-}^{(W)} &= \widetilde{x}_i - x_{i-}^{(W)} \end{aligned} \tag{10.17.10}$$

are the (asymmetric) distances from the confidence limits. If one takes $\chi_W^2(n_f) = 1$, the *asymmetric errors* Δx_{i+} and Δx_{i-} are obtained. The class MinAsy yields asymmetric errors or the distances from the confidence limits, respectively, for parameters, determined by minimization.

10.18 Examples

Example 10.1: Determining the parameters of a distribution from the elements of a sample with the method of maximum likelihood

Suppose one has N measurements y_1, y_2, …, y_n that can be assumed to come from a normal distribution with expectation value $a = x_1$ and standard deviation $\sigma = x_2$. The likelihood function is

$$L = \prod_{i=1}^{N} \frac{1}{x_2\sqrt{2\pi}} \exp\left\{-\frac{(y_i - x_1)^2}{2x_2^2}\right\} \tag{10.18.1}$$

and its logarithm is

$$\ell = -\sum_{i=1}^{N} \frac{(y_i - x_1)^2}{2x_2^2} - N \ln\{x_2\sqrt{2\pi}\} \quad . \tag{10.18.2}$$

The task of determining the maximum-likelihood estimators $\widetilde{x}_1, \widetilde{x}_2$ has already been solved in Example 7.8 by setting the analytically calculated derivatives of the function $\ell(\mathbf{x})$ to zero. Here we will accomplish this by means of numerical minimization of $\ell(\mathbf{x})$. For this we must first provide a function that computes the function to be minimized,

$$M(\mathbf{x}) = -\ell(\mathbf{x}) \quad .$$

This simple example is implemented in Example Programs 10.2 and 10.3. The results of the minimization of this user function are shown in Fig. 10.16 for two samples. Confidence regions and covariance ellipses agree reasonably well with each other. The agreement is better for the larger sample. ■

Example 10.2: Determination of the parameters of a distribution from the histogram of a sample by maximizing the likelihood

Instead of the original sample $y_1, y_2, \ldots, y_N$ as in Example 10.1, one often uses the corresponding histogram. Denoting by n_i the number of observations that fall into the interval centered about the point t_i with width Δt,

$$t_i - \Delta t/2 \le y < t_i + \Delta t/2 \quad , \tag{10.18.3}$$

the histogram is given by the pairs of numbers

$$(t_i, n_i) \quad , \qquad i = 1, 2, \ldots, n \quad . \tag{10.18.4}$$

If the original sample is taken from a normal distribution with expectation value $x_1 = a$ and standard deviation $x_2 = \sigma$, i.e., from the probability density

$$f(t; x_1, x_2) = \frac{1}{x_2\sqrt{2\pi}} \exp\left\{-\frac{(t - x_1)^2}{2x^2}\right\} \quad , \tag{10.18.5}$$

then one might expect the $n_i(t_i)$ (at least in the limit $N \to \infty$) to be given by

$$g_i = N \Delta t f(t_i; x_1, x_2) \quad . \tag{10.18.6}$$

The quantities $n_i(t_i)$ are integers, which are clearly not equal in general to g_i. We can, however, regard each $n_i(t_i)$ as a sample of size one from a Poisson distribution with the expectation value

$$\lambda_i = g_i \quad . \tag{10.18.7}$$

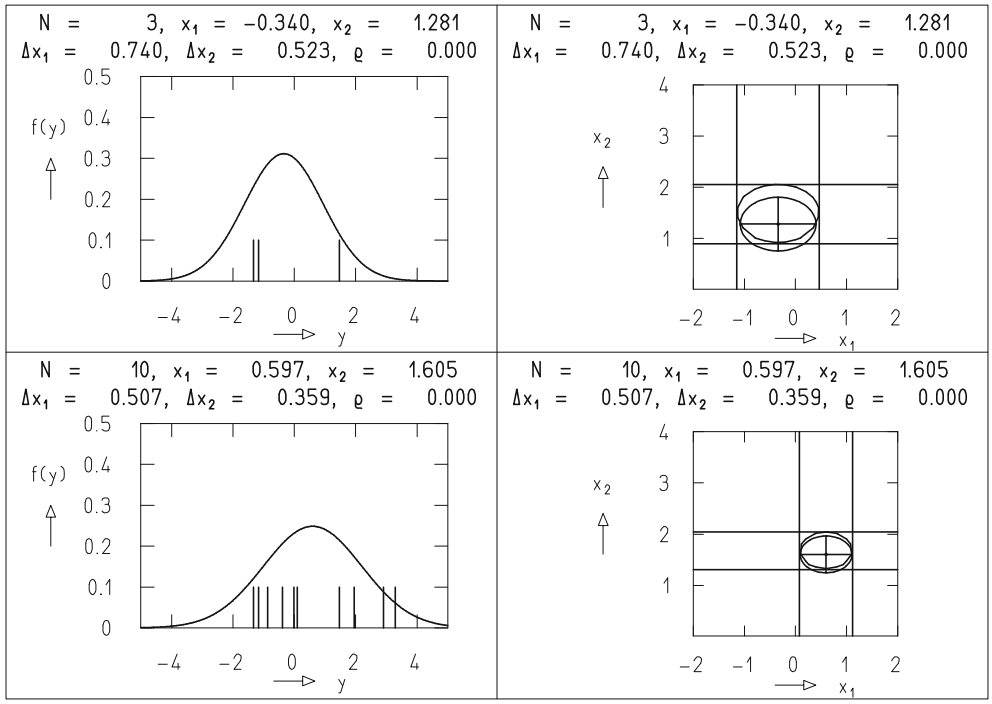

Fig. 10.16: Determination of the parameters x_1 (mean) and x_2 (width) of a Gaussian distribution by maximization of the log-likelihood function of a sample. The plots on the *left* show two different samples, marked as one-dimensional scatter plots (*tick marks*) on the y axis. The curves $f(y)$ are Gaussian distributions with the fitted parameters. The *right-hand plots* show in the (x_1, x_2) plane the values of the parameters obtained with symmetric errors and covariance ellipses, as well as the confidence region for $\chi^2_W = 1$ and the corresponding confidence boundaries (*horizontal* and *vertical lines*).

The a posteriori probability to observe the value $n_i(t_i)$ is clearly

$$\frac{1}{n_i!}\lambda_i^{n_i}\mathrm{e}^{-\lambda_i} \quad . \tag{10.18.8}$$

The likelihood function for the observation of the entire histogram is

$$L = \prod_{i=1}^{n} \frac{1}{n_i!}\lambda_i^{n_i}\mathrm{e}^{-\lambda_i} \quad , \tag{10.18.9}$$

and its logarithm is

$$\ell = -\sum_{i=1}^{n} \ln n_i! + \sum_{i=1}^{n} n_i \ln \lambda_i - \sum_{i=1}^{n} \lambda_i \quad . \tag{10.18.10}$$

If we use for λ_i the notation of (10.18.7) and find the minimum of $-\ell$ with respect to x_1, x_2, then this determines the best estimates of the parameters x_1, x_2. Of course the same procedure can be applied not only in the

case of a simple Gaussian distribution, but for any distribution that depends on parameters. One must simply use the corresponding probability density in (10.18.5) and (10.18.6). In the user function given below one must change only one instruction in order to replace the Gaussian by another distribution. This example is implemented in Example Program 10.4. There the user function carries the name MinLogLikeHistPoison.

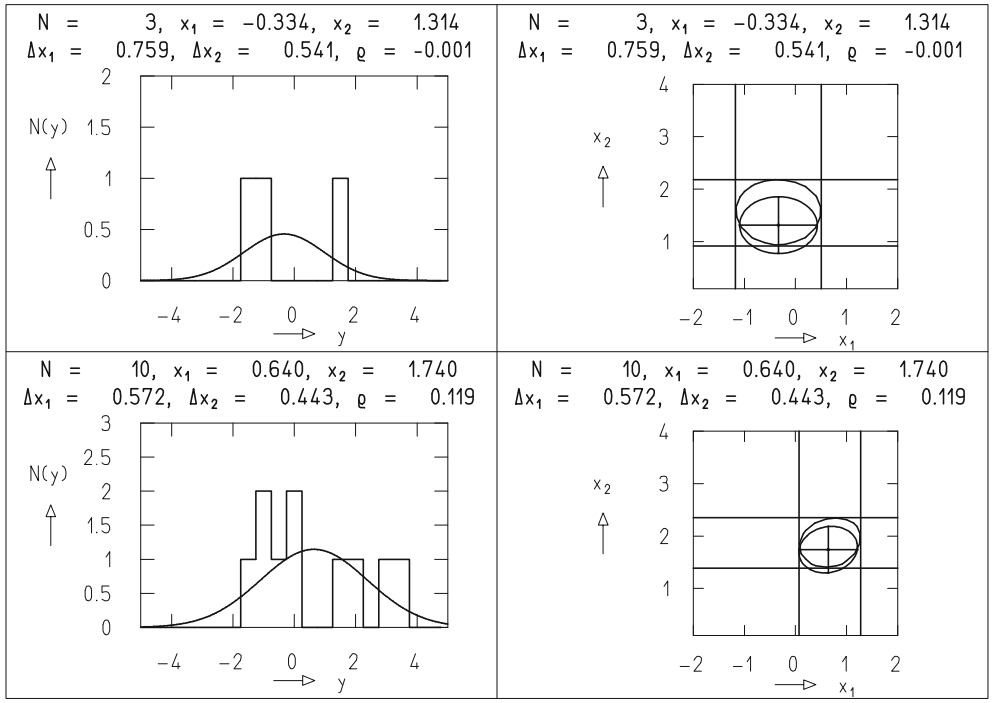

Fig. 10.17: Determination of the parameters x_1 (mean) and x_2 (width) of a Gaussian distribution by maximizing the log-likelihood function of a histogram. The *left-hand plots* show two histograms corresponding to the samples from Fig. 10.16. The curves are the Gaussian distributions normalized to the histograms. The *right-hand plots* show the symmetric errors, covariance ellipse, and confidence region represented as in Fig. 10.16.

The results of the minimization with this user function are shown in Fig. 10.17 for two histograms, which are based on the samples from Fig. 10.16. The results are very similar to those from Example 10.1. The errors of the parameters, however, are somewhat larger. This is to be expected, since some information is necessarily lost when constructing a histogram from the sample.

A histogram can be viewed as a sample in a compressed representation. The compression becomes greater as the bin width of the histogram increases. This is made clear in Fig. 10.18. One sees that for the same sample, the errors of the determined parameters increase for greater bin width. The effect is relatively small, however, for the relatively large sample size in this case. ■

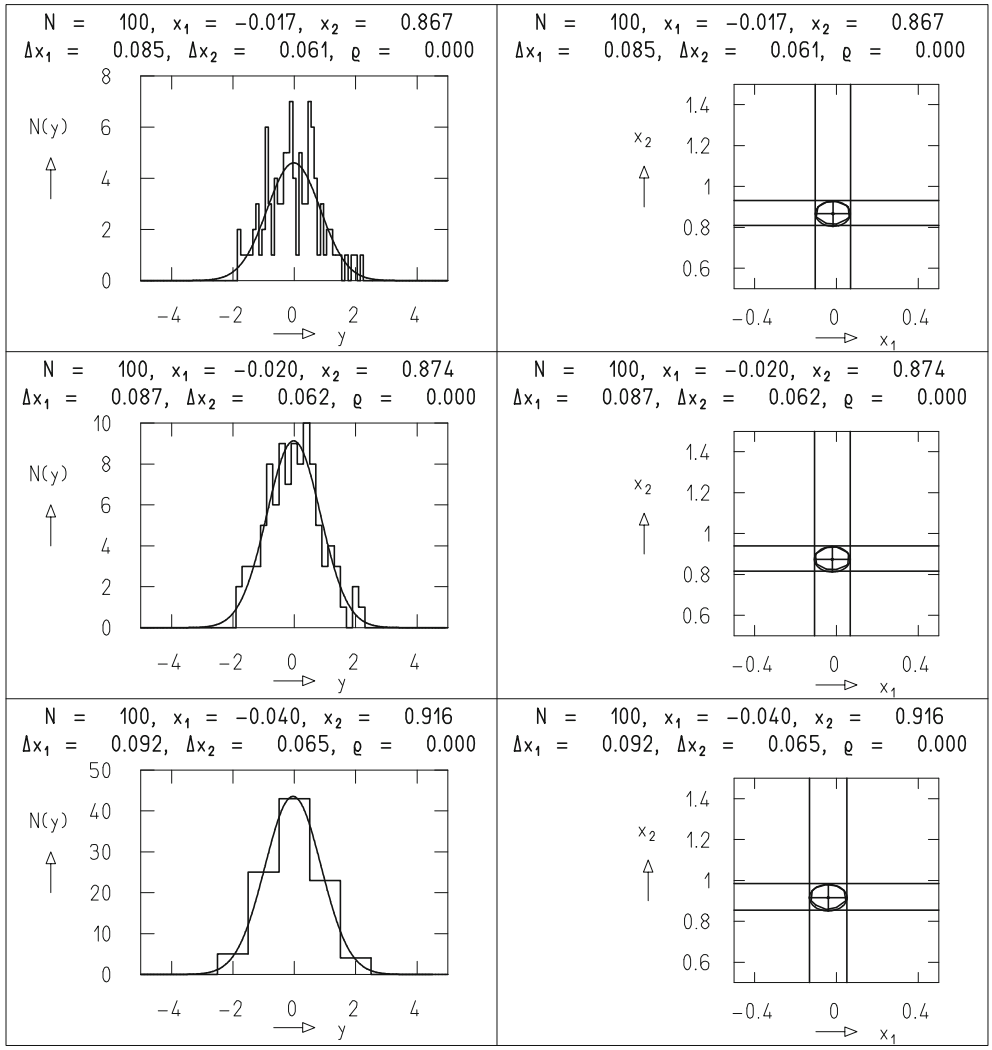

Fig. 10.18: As in Fig. 10.17, but for histograms with different interval widths of the same sample.

Example 10.3: Determination of the parameters of a distribution from the histogram of a sample by minimization of a sum of squares

If the bin contents n_i of a histogram (10.18.4) are sufficiently large, then the statistical fluctuations of each n_i can be approximated by a Gaussian distribution with the standard deviation

$$\Delta n_i = \sqrt{n_i} \quad . \tag{10.18.11}$$

(See Sect. 6.8.) The weighted sum of squares describing the deviation of the histogram contents n_i from the expected values g_i from (10.18.6) is then

$$Q = \sum_{i=1}^{N} \frac{(n_i - g_i)^2}{n_i} \quad . \tag{10.18.12}$$

When carrying out the sum one must take care (in contrast to Example 10.2) that empty bins ($n_i = 0$) are not included. Even better is to not include bins in the sum even with low bin contents, e.g., $n_i < 4$.

Also this example is implemented in Example Program 10.4. Here g_i is given by (10.18.6) as in the previous example. The sum is carried out over all bins with $n_i > 0$. The user function is called MinHistSumOfSquares.
The statistical fluctuations of the bin contents are taken as approximately Gaussian.

Figure 10.19 shows the results obtained by minimizing the quadratic sum for the histogram in Fig. 10.18. Since the histograms are based on the same sample, the values n_i decrease for decreasing bin width, whereby the requirement for using the sum of squares becomes less well fulfilled. Thus we cannot trust as much the results nor the errors for smaller bin width. One sees, however, that the errors given by the procedure in fact increase for decreasing bin width. ■

We emphasize that the determination of parameters from a histogram by quadratic-sum minimization gives less exact results than those obtained by likelihood maximization. This is because the assumption of a normal distribution of the n_i with the width (10.18.11) is only an approximation, which often requires large bin widths for the histogram and thus implies a loss of information. If, however, enough data, i.e., sufficiently large samples, are available, then the difference between the two procedures is small. One should compare, e.g., Fig. 10.18 (upper right-hand plot) with Fig. 10.19 (lower right-hand plot).

10.19 Java Classes and Example Programs

Java Classes for Minimization Problems

MinParab finds the extremum of a parabola through three given points.

FunctionOnline computes the value of a function on a straight line in n-dimensional space.

MinEnclose brackets the minimum on a straight line in n-dimensional space.

MinCombined finds the minimum in a given interval along a straight line with a combined method according tho Brent.

MinDir finds the minimum on a straight line in n-dimensional space.

MinSim finds the minimum in n-dimensional space using the simplex method.

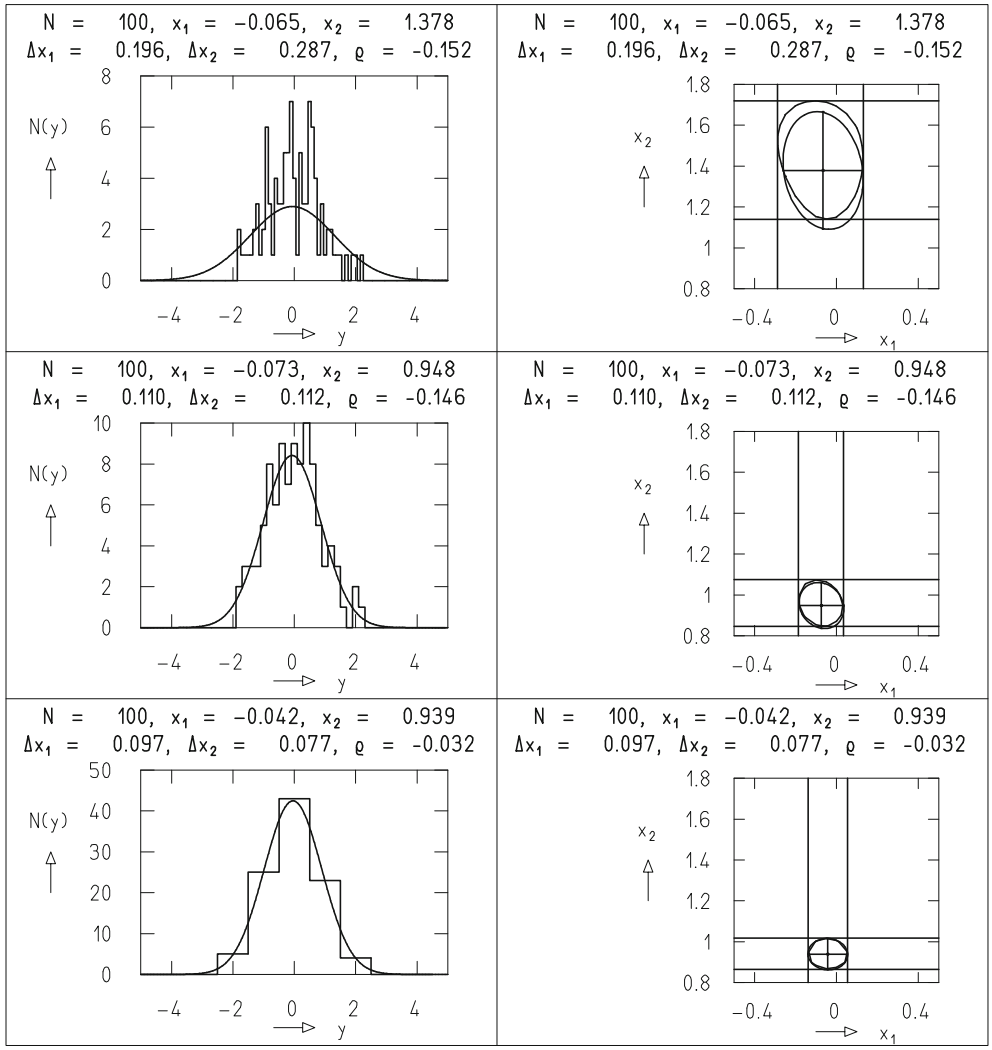

Fig. 10.19: Determination of the parameters x_1 (mean) and x_2 (width) of a Gaussian distribution by minimization of a weighted sum of squares. The histograms on the *left* are the same as in Fig. 10.18. The fit results and errors, covariance ellipses, and confidence regions are represented as in Figs. 10.17 and 10.18.

MinPow finds the minimum in n-dimensional space with Powell's method of chosen directions.

MinCjg finds the minimum in n-dimensional space with the method of conjugate directions.

MinQdr finds the minimum in n-dimensional space with quadratic form.

MInMar finds the minimum in n-dimensional space with Marquardt's method.

MinCov finds the covariance matrix of the coordinates of the minimum.

MinAsy finds the asymmetric errors of the coordinates of the minimum.

Example Program 10.1: The class E1Min demonstrates the use of MinSim, MinPow, MinCjg, MinQdr, and MinMar

The program calls one of the classes, as requested by the user, in order to solve the following problem. One wants to find the minimum of the function $f = f(\mathbf{x}) = f(x_1, x_2, x_3)$. The search is started at the point $\mathbf{x}^{(\mathrm{in})} = (x_1^{(\mathrm{in})}, x_2^{(\mathrm{in})}, x_3^{(\mathrm{in})})$, which is also input by the user. The program treats consecutively four different cases:

(i) No variables are fixed,

(ii) x_3 is fixed,

(iii) x_2 and x_3 are fixed,

(iv) All variables are fixed.

The user can choose one of the following functions to be minimized:

$$
\begin{aligned}
f_1(\mathbf{x}) &= r^2 \ , \quad r = \sqrt{x_1^2 + x_2^2 + x_3^2} \quad , \\
f_2(\mathbf{x}) &= r^{10} \ , \\
f_3(\mathbf{x}) &= r \ , \\
f_4(\mathbf{x}) &= -\mathrm{e}^{-r^2} \ , \\
f_5(\mathbf{x}) &= r^6 - 2r^4 + r^2 \ , \\
f_6(\mathbf{x}) &= r^2 \mathrm{e}^{-r^2} \ , \\
f_7(\mathbf{x}) &= -\mathrm{e}^{-r^2} - 10\,\mathrm{e}^{-r_a^2} \ , \quad r_a^2 = (x_1 - 3)^2 + (x_2 - 3)^2 + (x_3 - 3)^2 \ .
\end{aligned}
$$

Suggestions: Discuss the functions f_1 through f_7. All of them have a minimum at $x_1 = x_2 = x_3 = 0$. Some possess additional minima. Study the convergence behavior of the different minimization methods for these functions using different starting points and explain this behavior qualitatively.

Example Program 10.2: The class E2Min determines the parameters of a distribution from the elements of a sample and demonstrates the use of MinCov

The program solves the problem in Example 10.1. First a sample is drawn from the standard normal distribution. Next the sample is used to estimate the parameters x_1 (mean) and x_2 (standard deviation) of the population by minimizing the negative of the likelihood function (10.18.2). This is done withMinSim. The covariance matrix of the parameters is determined by callingMinCov. The results are presented numerically. The rest of the program performs the graphical display of the sample as a one-dimensional scatter plot and of the fitted function as in Fig. 10.16 (left-hand plots).

Suggestion: Run the program for samples of different size.

Example Program 10.3: The class E3Min demonstrates the use of MinAsy and draws the boundary of a confidence region

The class solves the same problem as the previous example. In addition it computes the asymmetric errors of the parameters using MinAsy. Then the solution, the symmetric errors, the covariance ellipse, and the asymmetric errors are displayed graphically in the (x_1, x_2) plane, and with the help of the method DatanGraphics.drawContour, the contour of the confidence region is shown as well. The plot corresponds to Fig. 10.16 (right-hand plots).

Example Program 10.4: The class E4Min determines the parameters of a distribution from the histogram of a sample

The program solves the problem of Examples 10.2 and 10.3. First a sample of size n_{ev} is drawn from a standard normal distribution and a histogram with n_t bins between $t_0 = -5.25$ and $t_{\max} = 5.25$ is constructed from the sample. The bin centers are t_i $(i = 1, 2, \ldots, n_t)$, and the bin contents are n_i. As chosen by the user either the likelihood function ℓ given by (10.18.10) is maximized (i.e., $-\ell$ is minimized) by MinSim and the user function MinLogLikeHistPoison, or the sum of squares Q given by (10.18.12) is minimized, again with MinSim, but using MinHistSumOfSquares.

The results are presented in graphical form. One plot contains the histogram and the fitted Gaussian (i.e., a plot corresponding to the plots on the left-hand side of Figs. 10.17 or 10.18), a second one presents the solution in the (x_1, x_2) plane with symmetric and asymmetric errors, covariance ellipse, and confidence region (corresponding to the plots on the right-hand side of those figures).

Suggestions: (a) Choose $n_{\text{ev}} = 100$. Show that for likelihood maximization the errors Δx_1, Δx_2 increase if you decrease the number of bins beginning with $n_t = 100$, but that for the minimization of the sum of squares the number of bins has to be small to get meaningful errors. (b) Show that for $n_{\text{ev}} = 1000$ and $n_t = 50$ or $n_t = 20$ there is practically no difference between the results of the methods.