Chapter 6
Roots and Extremal Points

In computational physics very often roots of a function, i.e. solutions of an equation like

$$f(x_1 \cdots x_N) = 0 \tag{6.1}$$

have to be determined. A related problem is the search for local extrema (Fig. 6.1)

$$\max f(x_1 \cdots x_N) \quad \min f(x_1 \cdots x_N) \tag{6.2}$$

which for a smooth function are solutions of the equations

$$\frac{\partial f(x_1 \cdots x_N)}{\partial x_i} = 0, \quad i = 1 \dots N. \tag{6.3}$$

In one dimension bisection is a very robust but rather inefficient root finding method. If a good starting point close to the root is available and the function smooth enough, the Newton–Raphson method converges much faster. Special strategies are necessary to find roots of not so well behaved functions or higher order roots. The combination of bisection and interpolation like in Dekker's and Brent's methods provides generally applicable algorithms. In multidimensions calculation of the Jacobian matrix is not always possible and Quasi-Newton methods are a good choice. Whereas local extrema can be found as the roots of the gradient, at least in principle, direct optimization can be more efficient. In one dimension the ternary search method or Brent's more efficient golden section search method can be used. In multidimensions the class of direction set search methods is very popular which includes the methods of steepest descent and conjugate gradients, the Newton–Raphson method and, if calculation of the full Hessian matrix is too expensive, the Quasi-Newton methods.

© Springer International Publishing AG 2017

P.O.J. Scherer, *Computational Physics*, Graduate Texts in Physics,
DOI 10.1007/978-3-319-61088-7_6

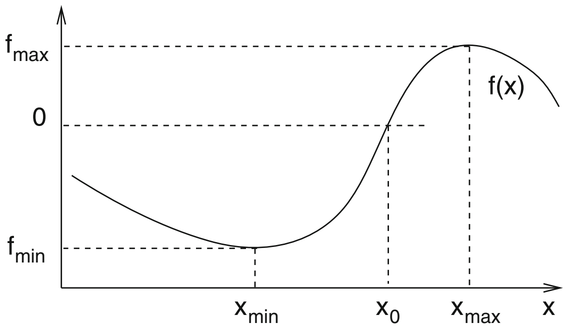


Fig. 6.1 Roots and local extrema of a function

6.1 Root Finding

If there is exactly one root in the interval $a_0 < x < b_0$ then one of the following methods can be used to locate the position with sufficient accuracy. If there are multiple roots, these methods will find one of them and special care has to be taken to locate the other roots.

6.1.1 Bisection

The simplest method [51] to solve

$$f(x) = 0 \tag{6.4}$$

uses the following algorithm (Fig. 6.2):

(1) Determine an interval $[a_0, b_0]$, which contains a sign change of $f(x)$. If no such interval can be found then $f(x)$ does not have any zero crossings
(2) Divide the interval into $[a_0, a_0 + \frac{b_0-a_0}{2}]$ $[a_0 + \frac{b_0-a_0}{2}, b_0]$ and choose that interval $[a_1, b_1]$, where $f(x)$ changes its sign.
(3) repeat until the width $b_n - a_n < \varepsilon$ is small enough.[1]

The bisection method needs two starting points which bracket a sign change of the function. It converges but only slowly since each step reduces the uncertainty by a factor of 2.

[1]Usually a combination like $\varepsilon = 2\varepsilon_M + |b_n|\varepsilon_r$ of an absolute and a relative tolerance is taken.

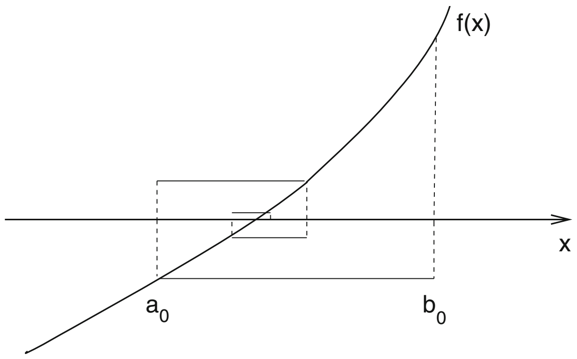


Fig. 6.2 Root finding by bisection

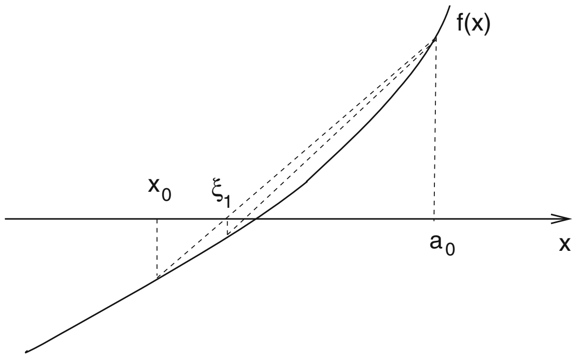


Fig. 6.3 Regula falsi method

6.1.2 Regula Falsi (False Position) Method

The regula falsi [52] method (Fig. 6.3) is similar to the bisection method [51]. However, polynomial interpolation is used to divide the interval $[x_r, a_r]$ with $f(x_r)f(a_r) < 0$. The root of the linear polynomial

$$p(x) = f(x_r) + (x - x_r)\frac{f(a_r) - f(x_r)}{a_r - x_r} \tag{6.5}$$

is given by

$$\xi_r = x_r - f(x_r)\frac{a_r - x_r}{f(a_r) - f(x_r)} = \frac{a_r f(x_r) - x_r f(a_r)}{f(x_r) - f(a_r)} \tag{6.6}$$

which is inside the interval $[x_r, a_r]$. Choose the sub-interval which contains the sign change:

$$\begin{aligned} f(x_r)f(\xi_r) < 0 &\rightarrow [x_{r+1}, a_{r+1}] = [x_r, \xi_r] \\ f(x_r)f(\xi_r) > 0 &\rightarrow [x_{r+1}, a_{r+1}] = [\xi_r, a_r]. \end{aligned} \tag{6.7}$$

Then ξ_r provides a series of approximations with increasing precision to the root of $f(x) = 0$.

6.1.3 Newton–Raphson Method

Consider a function which is differentiable at least two times around the root ξ. Taylor series expansion around a point x_0 in the vicinity

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0) + \cdots \tag{6.8}$$

gives for $x = \xi$

$$0 = f(x_0) + (\xi - x_0)f'(x_0) + \frac{1}{2}(\xi - x_0)^2 f''(x_0) + \cdots . \tag{6.9}$$

Truncation of the series and solving for ξ gives the first order Newton–Raphson [51, 53] method (Fig. 6.4)

$$x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)} \tag{6.10}$$

and the second order Newton–Raphson method (Fig. 6.4)

$$x_{r+1} = x_r - \frac{f'(x_r) \pm \sqrt{f'(x_r)^2 - 2f(x_r)f''(x_r)}}{f''(x_r)}. \tag{6.11}$$

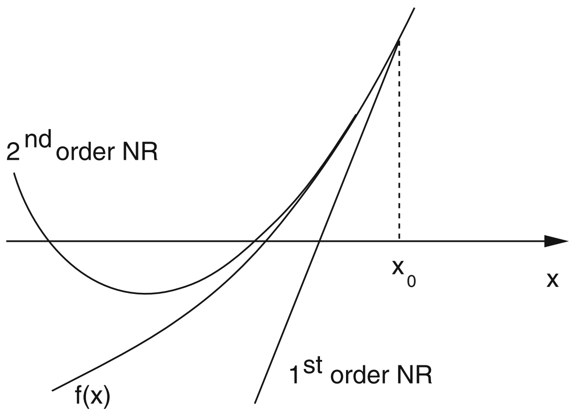
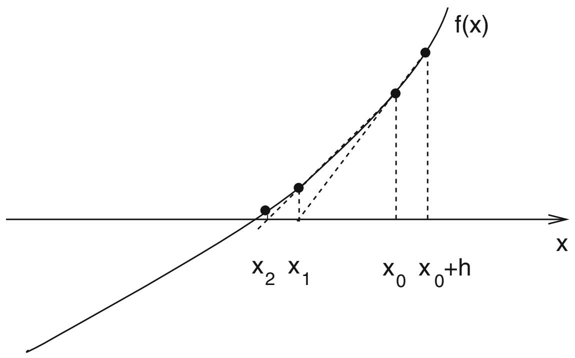


Fig. 6.4 Newton–Raphson method

Fig. 6.5 Secant method



The Newton–Raphson method converges fast if the starting point is close enough to the root. Analytic derivatives are needed. It may fail if two or more roots are close by.

6.1.4 Secant Method

Replacing the derivative in the first order Newton Raphson method by a finite difference quotient gives the secant method [51] (Fig. 6.5) which has been known for thousands of years before [54]

$$x_{r+1} = x_r - f(x_r)\frac{x_r - x_{r-1}}{f(x_r) - f(x_{r-1})}. \tag{6.12}$$

Round-off errors can become important as $|f(x_r) - f(x_{r-1})|$ gets small. At the beginning choose a starting point x_0 and determine

$$x_1 = x_0 - f(x_0)\frac{2h}{f(x_0 + h) - f(x_0 - h)} \tag{6.13}$$

using a symmetrical difference quotient.

6.1.5 Interpolation

The secant method is also obtained by linear interpolation

$$p(x) = \frac{x - x_r}{x_{r-1} - x_r} f_{r-1} + \frac{x - x_{r-1}}{x_r - x_{r-1}} f_r. \tag{6.14}$$

The root of the polynomial $p(x_{r+1}) = 0$ determines the next iterate x_{r+1}

$$x_{r+1} = \frac{1}{f_{r-1} - f_r}(x_r f_{r-1} - x_{r-1} f_r) = x_r - f_r \frac{x_r - x_{r-1}}{f_r - f_{r-1}}. \tag{6.15}$$

Quadratic interpolation of three function values is known as Muller's method [55]. Newton's form of the interpolating polynomial is

$$p(x) = f_r + (x - x_r) f[x_r, x_{r-1}] + (x - x_r)(x - x_{r-1}) f[x_r, x_{r-1}, x_{r-2}] \tag{6.16}$$

which can be rewritten

$$\begin{aligned} p(x) &= f_r + (x - x_r) f[x_r, x_{r-1}] + (x - x_r)^2 f[x_r, x_{r-1}, x_{r-2}] \\ &+ (x_r - x_{r-1})(x - x_r) f[x_r, x_{r-1}, x_{r-2}] \\ &= f_r + (x - x_r)^2 f[x_r, x_{r-1}, x_{r-2}] + (x - x_r)\left(f[x_r, x_{r-1}] + f[x_r, x_{r-2}] - f[x_{r-1}, x_{r-2}]\right) \\ &= f_r + A(x - x_r) + B(x - x_r)^2 \end{aligned} \tag{6.17}$$

and has the roots

$$x_{r+1} = x_r - \frac{A}{2B} \pm \sqrt{\frac{A^2}{4B^2} - \frac{f_r}{B}}. \tag{6.18}$$

To avoid numerical cancellation, this is rewritten

$$\begin{aligned} x_{r+1} &= x_r + \frac{1}{2B}\left(-A \pm \sqrt{A^2 - 4Bf_r}\right) \\ &= x_r + \frac{-2f_r}{A^2 - (A^2 - 4Bf_r)}\left(A \mp \sqrt{A^2 - 4Bf_r}\right) \\ &= x_r + \frac{-2f_r}{A \pm \sqrt{A^2 - 4Bf_r}}. \end{aligned} \tag{6.19}$$

The sign in the denominator is chosen such that x_{r+1} is the root closer to x_r. The roots of the polynomial can become complex valued and therefore this method is useful to find complex roots.

6.1.6 Inverse Interpolation

Complex values of x_r can be avoided by interpolation of the inverse function instead

$$x = f^{-1}(y). \tag{6.20}$$

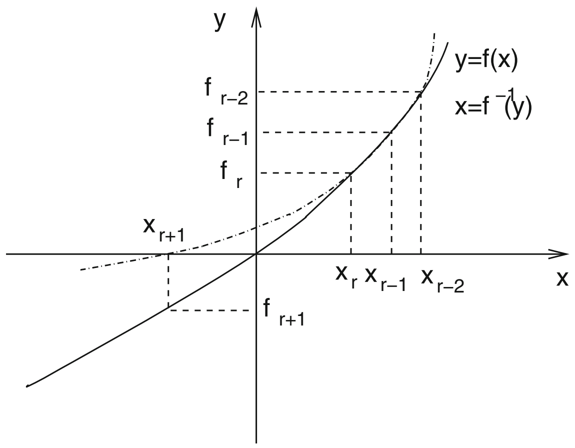


Fig. 6.6 Root finding by interpolation of the inverse function

Using the two points x_r, x_{r-1} the Lagrange method gives

$$p(y) = x_{r-1}\frac{y - f_r}{f_{r-1} - f_r} + x_r\frac{y - f_{r-1}}{f_r - f_{r-1}} \tag{6.21}$$

and the next approximation of the root corresponds again to the secant method (6.12)

$$x_{r+1} = p(0) = \frac{x_{r-1}f_r - x_r f_{r-1}}{f_r - f_{r-1}} = x_r + \frac{(x_{r-1} - x_r)}{f_r - f_{r-1}} f_r. \tag{6.22}$$

Inverse quadratic interpolation needs three starting points x_r, x_{r-1}, x_{r-2} together with the function values f_r, f_{r-1}, f_{r-2} (Fig. 6.6). The inverse function $x = f^{-1}(y)$ is interpolated with the Lagrange method

$$p(y) = \frac{(y - f_{r-1})(y - f_r)}{(f_{r-2} - f_{r-1})(f_{r-2} - f_r)}x_{r-2} + \frac{(y - f_{r-2})(y - f_r)}{(f_{r-1} - f_{r-2})(f_{r-1} - f_r)}x_{r-1}$$
$$+ \frac{(y - f_{r-1})(y - f_{r-2})}{(f_r - f_{r-1})(f_r - f_{r-2})}x_r. \tag{6.23}$$

For $y = 0$ we find the next iterate

$$x_{r+1} = p(0) = \frac{f_{r-1}f_r}{(f_{r-2} - f_{r-1})(f_{r-2} - f_r)}x_{r-2} + \frac{f_{r-2}f_r}{(f_{r-1} - f_{r-2})(f_{r-1} - f_r)}x_{r-1}$$
$$+ \frac{f_{r-1}f_{r-2}}{(f_r - f_{r-1})(f_r - f_{r-2})}x_r. \tag{6.24}$$

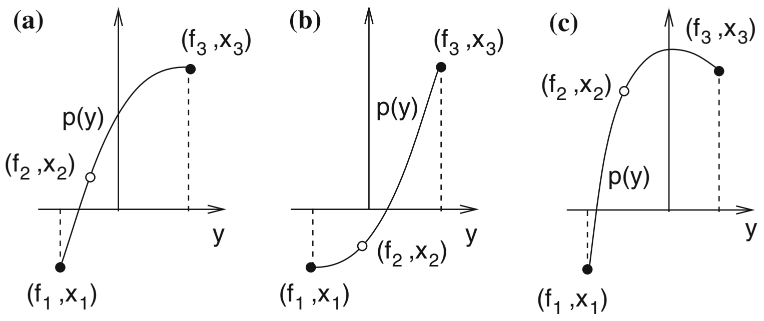


Fig. 6.7 (Validity of inverse quadratic interpolation) Inverse quadratic interpolation is only applicable if the interpolating polynomial $p(y)$ is monotonous in the range of the interpolated function values $f_1 \ldots f_3$. (**a**) and (**b**) show the limiting cases where the polynomial has a horizontal tangent at f_1 or f_3. (**c**) shows the case where the extremum of the parabola is inside the interpolation range and interpolation is not feasible

Inverse quadratic interpolation is only a good approximation if the interpolating parabola is single valued and hence if it is a monotonous function in the range of f_r, f_{r-1}, f_{r-2}. For the following discussion we assume that the three values of x are renamed such that $x_1 < x_2 < x_3$.

Consider the limiting case (a) in Fig. 6.7 where the polynomial has a horizontal tangent at $y = f_3$ and can be written as

$$p(y) = x_3 + (x_1 - x_3)\frac{(y - f_3)^2}{(f_1 - f_3)^2}. \tag{6.25}$$

Its value at $y = 0$ is

$$p(0) = x_3 + (x_1 - x_3)\frac{f_3^2}{(f_1 - f_3)^2} = x_1 + (x_3 - x_1)\left(1 - \frac{f_3^2}{(f_1 - f_3)^2}\right). \tag{6.26}$$

If f_1 and f_3 have different sign and $|f_1| < |f_3|$ (Sect. 6.1.7.2) we find

$$1 - \frac{f_3^2}{(f_1 - f_3)^2} < \frac{3}{4}. \tag{6.27}$$

Brent [56] used this as a criterion for the applicability of the inverse quadratic interpolation. However, this does not include all possible cases where interpolation is applicable. Chandrupatla [57] gave a more general discussion. The limiting condition is that the polynomial $p(y)$ has a horizontal tangent at one of the boundaries $x_{1,3}$. The derivative values are

$$\frac{dp}{dy}(y=f_1)=\frac{x_2(f_1-f_3)}{(f_2-f_1)(f_2-f_3)}+\frac{x_3(f_1-f_2)}{(f_3-f_1)(f_3-f_2)}+\frac{x_1}{f_1-f_2}+\frac{x_1}{f_1-f_3} \tag{6.28}$$

$$=\frac{(f_2-f_1)}{(f_3-f_1)(f_3-f_2)}\left[\frac{x_2(f_3-f_1)^2}{(f_2-f_1)^2}-x_3-\frac{x_1(f_3-f_1)^2-x_1(f_2-f_1)^2}{(f_2-f_1)^2}\right]$$

$$=\frac{(f_2-f_1)(x_2-x_1)}{(f_3-f_1)(f_3-f_2)}\left[\Phi^{-2}-\xi^{-1}\right]$$

$$\frac{dp}{dy}(y=f_3)=\frac{x_2(f_3-f_1)}{(f_2-f_1)(f_2-f_3)}+\frac{x_1(f_3-f_2)}{(f_1-f_2)(f_1-f_3)}+\frac{x_3}{f_3-f_2}+\frac{x_3}{f_3-f_1} \tag{6.29}$$

$$=\frac{(f_3-f_2)}{(f_2-f_1)(f_3-f_1)}\left[-\frac{x_2(f_3-f_1)^2}{(f_3-f_2)^2}+x_3\frac{(f_3-f_1)^2}{(f_3-f_2)^2}-x_3\frac{(f_3-f_2)^2}{(f_3-f_2)^2}+x_1\right]$$

$$=\frac{(f_3-f_2)(x_3-x_2)}{(f_2-f_1)(f_3-f_1)}\left[\left(\frac{1}{\Phi-1}\right)^2-\frac{1}{1-\xi}\right]$$

with [57]

$$\xi=\frac{x_2-x_1}{x_3-x_1}\quad \Phi=\frac{f_2-f_1}{f_3-f_1} \tag{6.30}$$

$$\xi-1=\frac{x_2-x_3}{x_3-x_1}\quad \Phi-1=\frac{f_2-f_3}{f_3-f_1}. \tag{6.31}$$

Since for a parabola either $f_1 < f_2 < f_3$ or $f_1 > f_2 > f_3$ the conditions for applicability of inverse interpolation finally become

$$\Phi^2 < \xi \tag{6.32}$$

$$1-\xi > (1-\Phi)^2 \tag{6.33}$$

which can be combined into

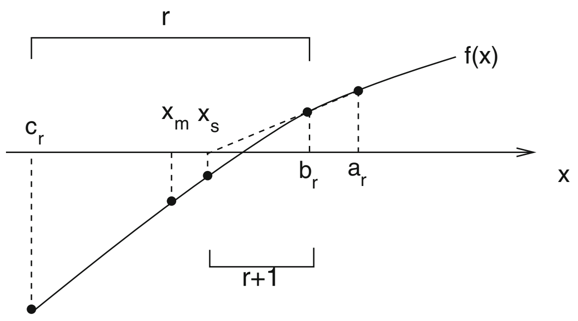
$$1-\sqrt{1-\xi} < |\Phi| < \sqrt{\xi}. \tag{6.34}$$

This method is usually used in combination with other methods (Sect. 6.1.7.2).

6.1.7 *Combined Methods*

Bisection converges slowly. The interpolation methods converge faster but are less reliable. The combination of both gives methods which are reliable and converge faster than pure bisection.

Fig. 6.8 Dekker's method



6.1.7.1 Dekker's Method

Dekker's method [58, 59] combines bisection and secant method. The root is bracketed by intervals $[c_r, b_r]$ with decreasing width where b_r is the best approximation to the root found and c_r is an earlier guess for which $f(c_r)$ and $f(b_r)$ have different sign. First an attempt is made to use linear interpolation between the points $(b_r, f(b_r))$ and $(a_r, f(a_r))$ where a_r is usually the preceding approximation $a_r = b_{r-1}$ and is set to the second interval boundary $a_r = c_{r-1}$ if the last iteration did not lower the function value (Fig. 6.8).

Starting from an initial interval $[x_0, x_1]$ with $\text{sign}(f(x_0)) \neq \text{sign}(f(x_1))$ the method proceeds as follows [59]:

initialization

```
$f_1 = f(x_1) \quad f_0 = f(x_0)$
if $|f_1| < |f_0|$ then {
$b = x_1 \quad c = a = x_0$
$f_b = f_1 \quad f_c = f_a = f_0$}
$else${
$b = x_0 \quad c = a = x_1$
$f_b = f_0 \quad f_c = f_a = f_1$}
```

iteration

```
$x_s = b - f_b \frac{b-a}{f_b - f_a}$
$x_m = \frac{c+b}{2}$.
```

If x_s is very close to the last b then increase the distance to avoid too small steps else choose x_s if it is between b and x_m, otherwise choose x_m (thus choosing the smaller interval)

$$x_r = \begin{cases} b + \delta \text{sign}(c - b) \quad \text{if abs}(x_s - b) < \delta \\ x_s \text{ if } b + \delta < x_s < x_m \text{ or } b - \delta > x_s > x_m \\ x_m \text{ else} \end{cases}.$$

Determine x_k as the latest of the previous iterates $x_0 \dots x_{r-1}$ for which $\mathrm{sign}(f(x_k)) \neq \mathrm{sign}(f(x_r))$.

If the new function value is lower update the approximation to the root

```
$f_r = f(x_r)$
if $|f_r| < |f_k|$ then {
$a = b \quad b = x_r \quad c = x_k$
$f_a = f_b \quad f_b = f_r \quad f_c = f_k$}
```

otherwise keep the old approximation and update the second interval boundary

```
if $|f_r| \geq |f_k|$ then {
$b = x_k \quad a = c = x_r$
$f_b = f_k \quad f_a = f_c = f_r$}
repeat until $|c - b| < \varepsilon$ or $f_r = 0$.
```

6.1.7.2 Brent's Method

In certain cases Dekker's method converges very slowly making many small steps of the order ϵ. Brent [56, 59, 60] introduced some modifications to reduce such problems and tried to speed up convergence by the use of inverse quadratic interpolation (Sect. 6.1.6). To avoid numerical problems the iterate (6.24) is written with the help of a quotient

$$\begin{aligned} x_{r+1} &= \frac{f_b f_c}{(f_a - f_b)(f_a - f_c)} a + \frac{f_a f_c}{(f_b - f_a)(f_b - f_c)} b \\ &+ \frac{f_b f_a}{(f_c - f_b)(f_c - f_a)} c \\ &= b + \frac{p}{q} \end{aligned} \tag{6.35}$$

with

$$\begin{aligned} p &= \frac{f_b}{f_a} \left((c - b) \frac{f_a}{f_c} \left(\frac{f_a}{f_c} - \frac{f_b}{f_c} \right) - (b - a) \left(\frac{f_b}{f_c} - 1 \right) \right) \\ &= (c - b) \frac{f_b (f_a - f_b)}{f_c^2} - (b - a) \frac{f_b (f_b - f_c)}{f_a f_c} \\ &= \frac{a f_b f_c (f_b - f_c) + b \left[f_a f_b (f_b - f_a) + f_b f_c (f_c - f_b) \right] + c f_a f_b (f_a - f_b)}{f_a f_c^2} \end{aligned} \tag{6.36}$$

$$q = -\left(\frac{f_a}{f_c} - 1\right)\left(\frac{f_b}{f_c} - 1\right)\left(\frac{f_b}{f_a} - 1\right) = -\frac{(f_a - f_c)(f_b - f_c)(f_b - f_a)}{f_a f_c^2}. \tag{6.37}$$

If only two points are available, linear interpolation is used. The iterate (6.22) then is written as

$$x_{r+1} = b + \frac{(a-b)}{f_b - f_a} f_b = b + \frac{p}{q} \tag{6.38}$$

with

$$p = (a-b)\frac{f_b}{f_a} \quad q = \left(\frac{f_b}{f_a} - 1\right). \tag{6.39}$$

The division is only performed if interpolation is appropriate and division by zero cannot happen. Brent's method is fast and robust at the same time. It is often recommended by text books. The algorithm is summarized in the following [61].
Start with an initial interval $[x_0, x_1]$ with $f(x_0)f(x_1) \le 0$

initialization

```
    $a = x_0 \quad b = x_1 \quad c = a$
    $f_a = f(a) \quad f_b = f(b) \quad f_c = f_a$
    $e = d = b - a$
```

iteration

If c is a better approximation than b exchange values

```
    if $|f_c| < |f_b|$ then{
    $a = b \quad b = c \quad c = a$
    $f_a = f_b \quad f_b = f_c \quad f_c = f_a$}
```

calculate midpoint relative to b

```
    $x_m = 0.5(c - b)$
```

stop if accuracy is sufficient

```
    if $|x_m| < \varepsilon$ or $f_b = 0$ then exit
```

use bisection if the previous step width e was too small or the last step did not improve

```
    if $|e| < \varepsilon$ or $|f_a| \le |f_b|$ then{
    $e = d = x_m$}
```

otherwise try interpolation

```
    else {
    if $a = c$ then {
    $p = 2x_m \frac{f_b}{f_a} \quad q = \frac{f_b - f_a}{f_a}$}
    else {
    $p = 2x_m \frac{f_b(f_a - f_b)}{f_c^2} - (b-a)\frac{f_b(f_b - f_c)}{f_a f_c}$
    $q = \left(\frac{f_a}{f_c} - 1\right)\left(\frac{f_b}{f_c} - 1\right)\left(\frac{f_b}{f_a} - 1\right)$}
```

make p a positive quantity

```
    if $p > 0$ then {$q = -q$} else {$p = -p$}

    update previous step width
    $s = e \quad e = d$
    use interpolation if applicable, otherwise use bisection

    if $2p < 3x_m q - |\varepsilon q|$ and $p < |0.5\, sq|$ then{
    $d = \frac{p}{q}$}
    else{$e = d = x_m$}
    $a = b \quad f_a = f_b$

    if $|d| > \varepsilon$ then {
    $b = b + d$}
    else {$\mathrm{b} = \mathrm{b} + \varepsilon \mathrm{sign}\ (x_m)$}

    calculate new function value
    $f_b = f(b)$
    be sure to bracket the root
    if sign($f_b$) = sign($f_c$) then {
    $c = a \quad f_c = f_a$
    $e = d = b - a$}
```

6.1.7.3 Chandrupatla's Method

In 1997 Chandrupatla [57] published a method which tries to use inverse quadratic interpolation whenever possible according to (6.34). He calculates the relative position of the new iterate as (Fig. 6.9).

$$t = \frac{x - c}{b - c}$$

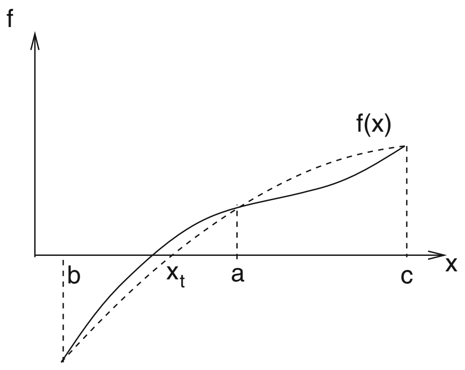


Fig. 6.9 Chandrupatla's method

$$= \frac{1}{b-c}\left[\frac{f_c f_b}{(f_a-f_b)(f_a-f_c)}a + \frac{f_a f_c}{(f_b-f_a)(f_b-f_c)}b + \frac{f_b f_a}{(f_c-f_b)(f_c-f_a)}c - c\right]$$
$$= \frac{a-c}{b-c}\frac{f_c}{f_c-f_a}\frac{f_b}{f_b-f_a} + \frac{f_a f_c}{(f_b-f_a)(f_b-f_c)}. \tag{6.40}$$

The algorithm proceeds as follows:
Start with an initial interval $[x_0, x_1]$ with $f(x_0)f(x_1) \le 0$.

```
initialization
$b = x_0 \quad a = c = x_1$
$f_b = f(b) \quad f_a = f_c = f(c)$
$t = 0.5$
iteration
$x_t = a + t\,(b-a)$
$f_t = f(x_t)$
if sign($f_t$) = sign($f_a$){
$c = a \quad f_c = f_a$
$a = x_t \quad f_a = F_t$}
else{
$c = b \quad b = a \quad a = x_t$
$f_c = f_b \quad f_b = f_a \quad f_a = f_t$}
$x_m = a \quad f_m = f_a$
if abs ($f_b$) < abs($f_a$){
$x_m = b \quad f_m = f_b$}
$tol = 2\epsilon_M|x_m| + \epsilon_a$
$t_l = \frac{tol}{|b-c|}$
```

```
if $t_l > 0.5$ or $f_m = 0$ exit
$\xi = \frac{a-b}{c-b} \quad \Phi = \frac{f_a-f_b}{f_c-f_b}$
if $1-\sqrt{1-\xi} < \Phi < \sqrt{\xi}$ {
$t = \frac{f_a}{f_b-f_a}\frac{f_c}{f_b-f_c} + \frac{c-a}{b-a}\frac{f_a}{f_c-f_a}\frac{f_b}{f_c-f_b}$ }
else { $t = 0.5$ }
if $t < t_l$ { $t = t_l$ }
if $t > (1-t_l)$ { $t = 1-t_l$ }
```

Chandrupatla's method is more efficient than Dekker's and Brent's, especially for higher order roots (Figs. 6.10, 6.11 and 6.12).

6.1.8 Multidimensional Root Finding

The Newton–Raphson method can be easily generalized for functions of more than one variable. We search for the solution of a system of n nonlinear equations in n variables x_i

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1 \cdots x_n) \\ \vdots \\ f_n(x_1 \cdots x_n) \end{pmatrix} = 0. \tag{6.41}$$

The first order Newton–Raphson method results from linearization of

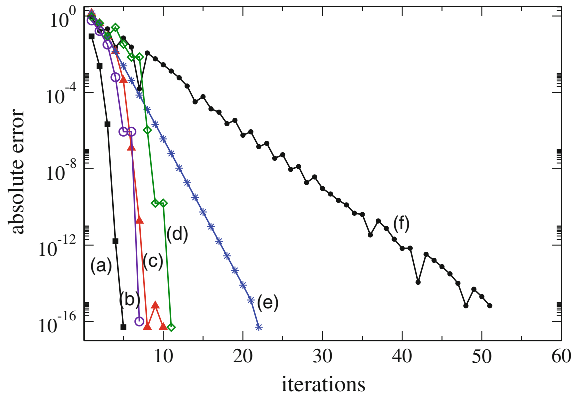


Fig. 6.10 (Comparison of different solvers) The root of the equation $f(x) = x^2 - 2$ is determined with different methods: Newton–Raphson (**a**) (*black squares*), Chandrupatla (**b**) (*indigo circles*), Brent (**c**) (*red triangles up*), Dekker (**d**) (*green diamonds*), regula falsi (**e**) (*blue stars*), pure bisection (**f**) (*black dots*). Starting values are $x_1 = -1, x_2 = 2$. The absolute error is shown as function of the number of iterations. For $x_1 = -1$, the Newton–Raphson method converges against $-\sqrt{2}$

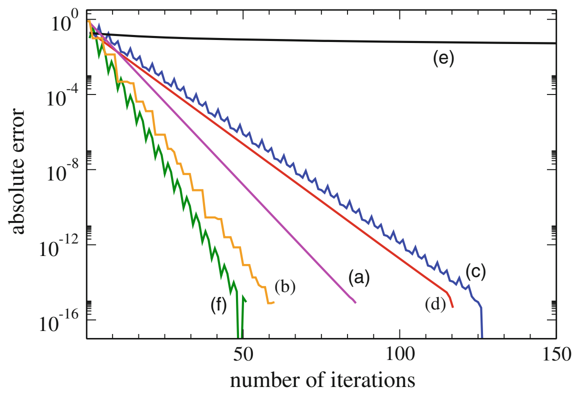


Fig. 6.11 (Comparison of different solvers for a third order root) The root of the equation $f(x) = (x-1)^3$ is determined with different methods: Newton–Raphson (**a**) (*magenta*), Chandrupatla (**b**) (*orange*), Brent (**c**) (*blue*), Dekker (**d**) (*red*), regula falsi (**e**) (*black*), pure bisection (**f**) (*green*). Starting values are $x_1 = 0, x_2 = 1.8$. The absolute error is shown as function of the number of iterations

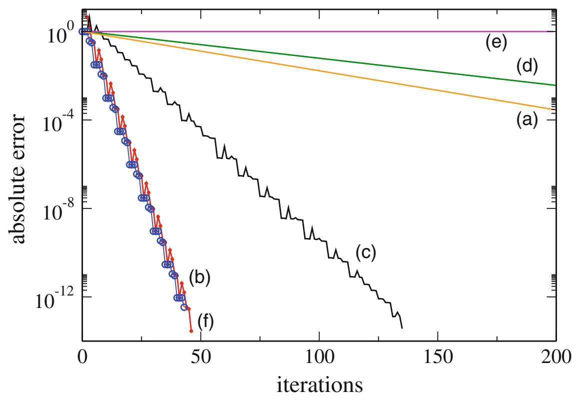


Fig. 6.12 (Comparison of different solvers for a high order root) The root of the equation $f(x) = x^{25}$ is determined with different methods: Newton–Raphson (**a**) (*orange*), Chandrupatla (**b**) (*blue circles*), Brent (**c**) (*black*), Dekker (**d**) (*green*), regula falsi (**e**) (*magenta*), pure bisection (**f**) (*red dots*). Starting values are $x_1 = -1, x_2 = 2$. The absolute error is shown as function of the number of iterations

$$0 = \mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}^0) + J(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) + \cdots \tag{6.42}$$

with the Jacobian matrix

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}. \tag{6.43}$$

If the Jacobian matrix is not singular the equation

$$0 = \mathbf{f}(\mathbf{x}^0) + J(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) \tag{6.44}$$

can be solved by

$$\mathbf{x} = \mathbf{x}^0 - \left(J(\mathbf{x}^0)\right)^{-1} \mathbf{f}(\mathbf{x}^0). \tag{6.45}$$

This can be repeated iteratively

$$\mathbf{x}^{(r+1)} = \mathbf{x}^{(r)} - (J(\mathbf{x}^{(r)}))^{-1}\mathbf{f}(\mathbf{x}^{(r)}). \tag{6.46}$$

6.1.9 Quasi-Newton Methods

Calculation of the Jacobian matrix can be very time consuming. Quasi-Newton methods use instead an approximation to the Jacobian which is updated during each iteration. Defining the differences

$$\mathbf{d}^{(r)} = \mathbf{x}^{(r+1)} - \mathbf{x}^{(r)} \tag{6.47}$$

$$\mathbf{y}^{(r)} = \mathbf{f}(\mathbf{x}^{(r+1)}) - \mathbf{f}(\mathbf{x}^{(r)}) \tag{6.48}$$

we obtain from the truncated Taylor series

$$\mathbf{f}(\mathbf{x}^{(r+1)}) = \mathbf{f}(\mathbf{x}^{(r)}) + J(\mathbf{x}^{(r)})(\mathbf{x}^{(r+1)} - \mathbf{x}^{(r)}) \tag{6.49}$$

the so called Quasi-Newton or secant condition

$$\mathbf{y}^{(r)} = J(\mathbf{x}^{(r)})\mathbf{d}^{(r)}. \tag{6.50}$$

We attempt to construct a family of successive approximation matrices J_r so that, if J were a constant, the procedure would become consistent with the quasi-Newton condition. Then for the new update J_{r+1} we have

$$J_{r+1}\mathbf{d}^{(r)} = \mathbf{y}^{(r)}. \tag{6.51}$$

Since $\mathbf{d}^{(r)}$, $\mathbf{y}^{(r)}$ are already known, these are only n equations for the n^2 elements of J_{r+1}. To specify J_{r+1} uniquely, additional conditions are required. For instance, it is reasonable to assume, that

$$J_{r+1}\mathbf{u} = J_r\mathbf{u} \quad \text{for all } \mathbf{u} \perp \mathbf{d}^{(r)}. \tag{6.52}$$

Then J_{r+1} differs from J_r only by a rank one updating matrix

$$J_{r+1} = J_r + \mathbf{u}\,\mathbf{d}^{(r)T}. \tag{6.53}$$

From the secant condition we obtain

$$J_{r+1}\mathbf{d}^{(r)} = J_r\mathbf{d}^{(r)} + \mathbf{u}(\mathbf{d}^{(r)}\mathbf{d}^{(r)T}) = \mathbf{y}^{(r)} \tag{6.54}$$

hence

$$\mathbf{u} = \frac{1}{|d^{(r)}|^2}\left(\mathbf{y}^{(r)} - J_r\mathbf{d}^{(r)}\right). \tag{6.55}$$

This gives Broyden's update formula [62]

$$J_{r+1} = J_r + \frac{1}{|d^{(r)}|^2}\left(\mathbf{y}^{(r)} - J_r\mathbf{d}^{(r)}\right)\mathbf{d}^{(r)T}. \tag{6.56}$$

To update the inverse Jacobian matrix, the Sherman–Morrison formula [42]

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}} \tag{6.57}$$

can be applied to have

$$\begin{aligned} J_{r+1}^{-1} &= J_r^{-1} - \frac{J_r^{-1}\frac{1}{|d^{(r)}|^2}\left(\mathbf{y}^{(r)} - J_r\mathbf{d}^{(r)}\right)\mathbf{d}^{(r)T}J_r^{-1}}{1 + \frac{1}{|d^{(r)}|^2}\mathbf{d}^{(r)T}J_r^{-1}\left(\mathbf{y}^{(r)} - J_r\mathbf{d}^{(r)}\right)} \\ &= J_r^{-1} - \frac{\left(J_r^{-1}\mathbf{y}^{(r)} - \mathbf{d}^{(r)}\right)\mathbf{d}^{(r)T}J_r^{-1}}{\mathbf{d}^{(r)T}J_r^{-1}\mathbf{y}^{(r)}}. \end{aligned} \tag{6.58}$$

6.2 Function Minimization

Minimization or maximization of a function[2] is a fundamental task in numerical mathematics and closely related to root finding. If the function $f(x)$ is continuously differentiable then at the extremal points the derivative is zero

$$\frac{\mathrm{d}f}{\mathrm{d}x} = 0. \tag{6.59}$$

Hence, in principle root finding methods can be applied to locate local extrema of a function. However, in some cases the derivative cannot be easily calculated or the

[2]In the following we consider only a minimum since a maximum could be found as the minimum of $-f(x)$.

function even is not differentiable. Then derivative free methods similar to bisection for root finding have to be used.

6.2.1 *The Ternary Search Method*

Ternary search is a simple method to determine the minimum of a unimodal function $f(x)$. Initially we have to find an interval $[a_0, b_0]$ which is certain to contain the minimum. Then the interval is divided into three equal parts $[a_0, c_0]$, $[c_0, d_0]$, $[d_0, b_0]$ and either the first or the last of the three intervals is excluded (Fig. 6.13). The procedure is repeated with the remaining interval $[a_1, b_1] = [a_0, d_0]$ or $[a_1, b_1] = [c_0, b_0]$.

Each step needs two function evaluations and reduces the interval width by a factor of 2/3 until the maximum possible precision is obtained. It can be determined by considering a differentiable function which can be expanded around the minimum x_0 as

$$f(x) = f(x_0) + \frac{(x-x_0)^2}{2} f''(x_0) + \cdots . \tag{6.60}$$

Numerically calculated function values $f(x)$ and $f(x_0)$ only differ, if

$$\frac{(x-x_0)^2}{2} f''(x_0) > \varepsilon_M f(x_0) \tag{6.61}$$

which limits the possible numerical accuracy to

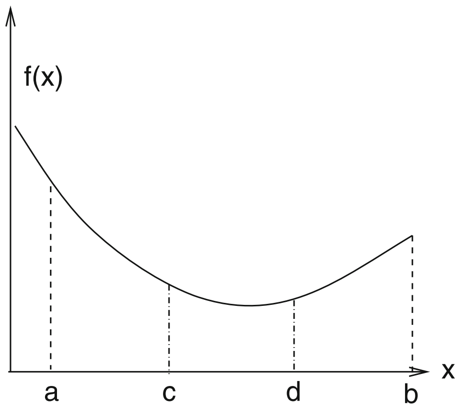


Fig. 6.13 Ternary search method

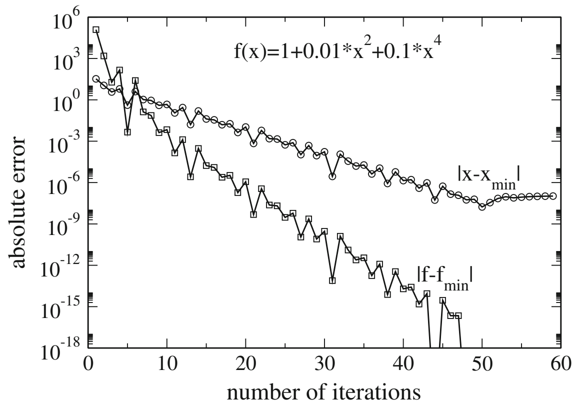


Fig. 6.14 (Ternary search method) The minimum of the function $f(x) = 1 + 0.01\,x^2 + 0.1\,x^4$ is determined with the ternary search method. Each iteration needs two function evaluations. After 50 iterations the function minimum $f_{\min} = 1$ is reached to machine precision $\varepsilon_M \approx 10^{-16}$. The position of the minimum $x_{\min}$ cannot be determined with higher precision than $\sqrt{\varepsilon_M} \approx 10^{-8}$ (6.63)

$$\varepsilon(x_0) = \min|x - x_0| = \sqrt{\frac{2f(x_0)}{f''(x_0)}\varepsilon_M} \tag{6.62}$$

and for reasonably well behaved functions (Fig. 6.14) we have the rule of thumb [63]

$$\varepsilon(x_0) \approx \sqrt{\varepsilon_M}. \tag{6.63}$$

However, it may be impossible to reach even this precision, if the quadratic term of the Taylor series vanishes (Fig. 6.15).

The algorithm can be formulated as follows:

iteration

```
if$(b - a) < \delta$ then exit
$c = a + \frac{1}{3}(b - a)$   $d = a + \frac{2}{3}(b - a)$
$f_c = f(c)$   $f_d = f(d)$
if $f_c < f_d$ then $b = d$ else $a = c$
```

6.2.2 *The Golden Section Search Method (Brent's Method)*

To bracket a local minimum of a unimodal function $f(x)$ three points a, b, c are necessary (Fig. 6.16) with

$$f(a) > f(b) \quad f(c) > f(b). \tag{6.64}$$

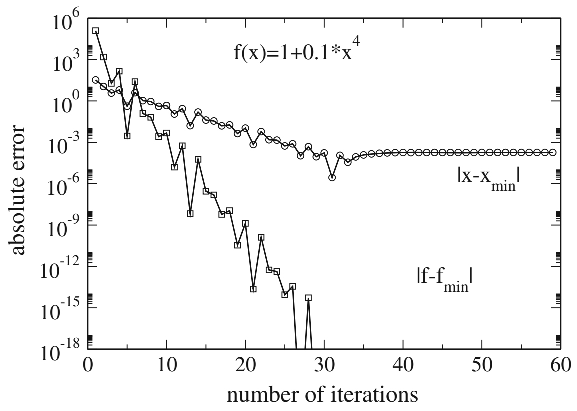


Fig. 6.15 (Ternary search method for a higher order minimum) The minimum of the function $f(x) = 1 + 0.1\,x^4$ is determined with the ternary search method. Each iteration needs two function evaluations. After 30 iterations the function minimum $f_{\min} = 1$ is reached to machine precision $\varepsilon_M \approx 10^{-16}$. The position of the minimum $x_{\min}$ cannot be determined with higher precision than $\sqrt[4]{\varepsilon_M} \approx 10^{-4}$

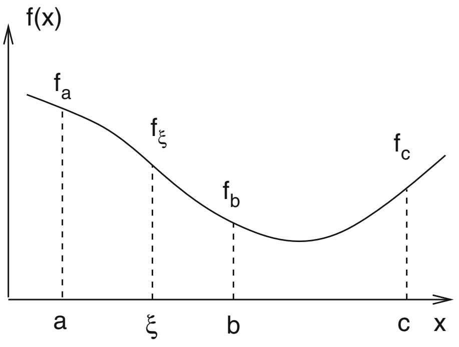


Fig. 6.16 (Golden section search method) A local minimum of the function $f(x)$ is bracketed by three points a, b, c. To reduce the uncertainty of the minimum position a new point ξ is chosen in the interval $a < \xi < c$ and either a or c is dropped according to the relation of the function values. For the example shown a has to be replaced by ξ

The position of the minimum can be determined iteratively by choosing a new value ξ in the interval $a < \xi < c$ and dropping either a or c, depending on the ratio of the function values. A reasonable choice for ξ can be found as follows (Fig. 6.17) [63, 64]. Let us denote the relative positions of the middle point and the trial point as

$$\frac{b-a}{c-a} = \beta \quad \frac{c-b}{c-a} = 1-\beta \quad \frac{b-a}{c-b} = \frac{\beta}{1-\beta} \quad \frac{\xi-b}{c-a} = t.$$
$$\frac{\xi-a}{c-a} = \frac{\xi-b+b-a}{c-a} = t+\beta. \tag{6.65}$$

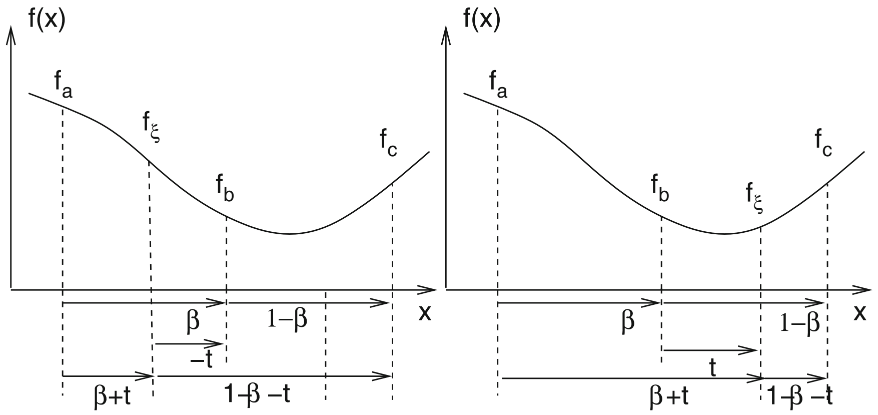


Fig. 6.17 Golden section search method

The relative width of the new interval will be

$$\frac{c-\xi}{c-a} = (1-\beta-t) \quad \text{or} \quad \frac{b-a}{c-a} = \beta \quad \text{if } a < \xi < b \tag{6.66}$$

$$\frac{\xi-a}{c-a} = (t+\beta) \quad \text{or} \quad \frac{c-b}{c-a} = (1-\beta) \quad \text{if } b < \xi < c. \tag{6.67}$$

The golden search method requires that

$$t = 1-2\beta = \frac{c+a-2b}{c-a} = \frac{(c-b)-(b-a)}{c-a}. \tag{6.68}$$

Otherwise it would be possible that the larger interval width is selected many times slowing down the convergence. The value of t is positive if $c-b > b-a$ and negative if $c-b < b-a$, hence the trial point always is in the larger of the two intervals. In addition the golden search method requires that the ratio of the spacing remains constant. Therefore we set

$$\frac{\beta}{1-\beta} = -\frac{t+\beta}{t} = -\frac{1-\beta}{t} \quad \text{if } a < \xi < b \tag{6.69}$$

$$\frac{\beta}{1-\beta} = \frac{t}{1-\beta-t} = \frac{t}{\beta} \quad \text{if } b < \xi < c. \tag{6.70}$$

Eliminating t we obtain for $a < \xi < b$ the equation

$$\frac{(\beta-1)}{\beta}(\beta^2+\beta-1) = 0. \tag{6.71}$$

Besides the trivial solution $\beta = 1$ there is only one positive solution

$$\beta = \frac{\sqrt{5}-1}{2} \approx 0.618. \tag{6.72}$$

For $b < \xi < c$ we end up with

$$\frac{\beta}{\beta - 1}(\beta^2 - 3\beta + 1) = 0 \tag{6.73}$$

which has the nontrivial positive solution

$$\beta = \frac{3-\sqrt{5}}{2} \approx 0.382. \tag{6.74}$$

Hence the lengths of the two intervals $[a, b]$, $[b, c]$ have to be in the golden ratio $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ which gives the method its name. Using the golden ratio the width of the interval bracketing the minimum reduces by a factor of 0.618 (Figs. 6.18 and 6.19).

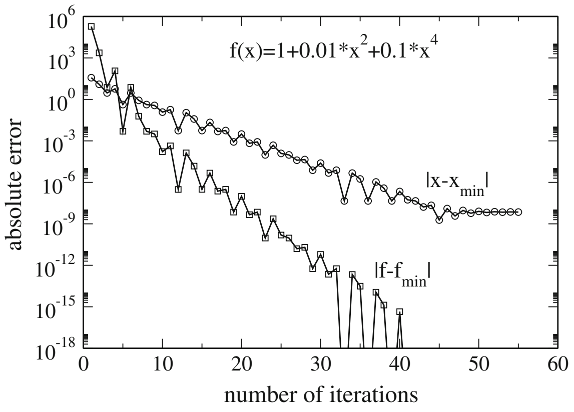


Fig. 6.18 (Golden section search method) The minimum of the function $f(x) = 1 + 0.01\,x^2 + 0.1\,x^4$ is determined with the golden section search method. Each iteration needs only one function evaluation. After 40 iterations the function minimum $f_{\min} = 1$ is reached to machine precision $\varepsilon_M \approx 10^{-16}$. The position of the minimum $x_{\min}$ cannot be determined to higher precision than $\sqrt{\varepsilon_M} \approx 10^{-8}$ (6.63)

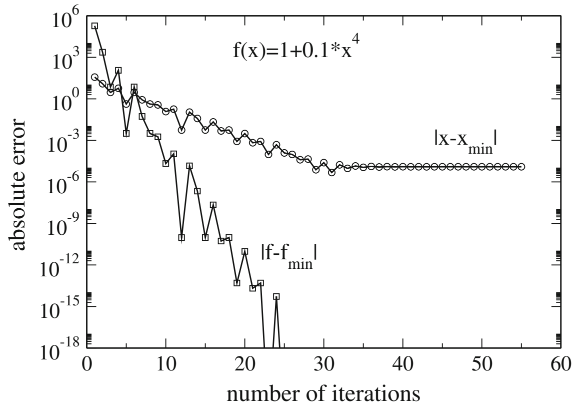


Fig. 6.19 (Golden section search for a higher order minimum) The minimum of the function $f(x) = 1 + 0.1\,x^4$ is determined with the golden section search method. Each iteration needs only one function evaluation. After 28 iterations the function minimum $f_{\min} = 1$ is reached to machine precision $\varepsilon_M \approx 10^{-16}$. The position of the minimum $x_{\min}$ cannot be determined to higher precision than $\sqrt[4]{\varepsilon_M} \approx 10^{-4}$

The algorithm can be formulated as follows:

```
if $c - a < \delta$ then exit
if $(b - a) \geq (c - b)$ then {
$x = 0.618\,b + 0.382\,a$
$f_x = f(x)$
if $f_x < f_b$ then {$c = b \quad b = x \quad f_c = f_b \quad f_b = f_x$}
else $a = x \quad f_a = f_x$}
if $(b - a) < (c - b)$ then {
$x = 0.618\,b + 0.382\,c$
$f_x = f(x)$
if $f_x < f_b$ then {$a = b \quad b = x \quad f_a = f_b \quad f_b = f_x$}
else $c = x \quad f_c = f_x$}
```

To start the method we need three initial points which can be found by Brent's exponential search method (Fig. 6.20). Begin with three points

$$a_0, b_0 = a_0 + h, c_0 + 1.618\,h \tag{6.75}$$

where h_0 is a suitable initial step width which depends on the problem. If the minimum is not already bracketed then if necessary exchange a_0 and b_0 to have

$$f(a_0) > f(b_0) > f(c_0). \tag{6.76}$$

Then replace the three points by

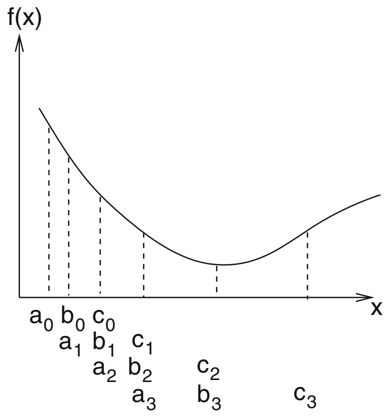


Fig. 6.20 Brent's exponential search

$$a_1 = b_0 \quad b_1 = c_0 \quad c_1 = c_0 + 1.618\,(c_0 - b_0) \tag{6.77}$$

and repeat this step until

$$f(b_n) < f(c_n) \tag{6.78}$$

or n exceeds a given maximum number. In this case no minimum can be found and we should check if the initial step width was too large.

Brent's method can be improved by making use of derivatives and by combining the golden section search with parabolic interpolation [63].

6.2.3 Minimization in Multidimensions

We search for local minima (or maxima) of a function

$$h(\mathbf{x})$$

which is at least two times differentiable. In the following we denote the gradient vector by

$$\mathbf{g}^T(\mathbf{x}) = \left(\frac{\partial h}{\partial x_1}, \cdots \frac{\partial h}{\partial x_n}\right) \tag{6.79}$$

and the matrix of second derivatives (Hessian) by

$$H = \left(\frac{\partial^2}{\partial x_i \partial x_j} h\right). \tag{6.80}$$

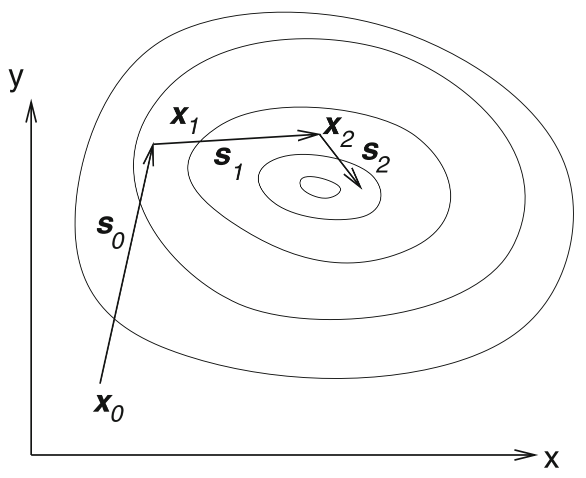


Fig. 6.21 (Direction set minimization) Starting from an initial guess $\mathbf{x}_0$ a local minimum is approached by making steps along a set of direction vectors $\mathbf{s}_r$

The very popular class of direction set methods proceeds as follows (Fig. 6.21). Starting from an initial guess $\mathbf{x}_0$ a set of direction vectors $\mathbf{s}_r$ and step lengths λ_r is determined such that the series of vectors

$$\mathbf{x}_{r+1} = \mathbf{x}_r + \lambda_r \mathbf{s}_r \tag{6.81}$$

approaches the minimum of $h(\mathbf{x})$. The method stops if the norm of the gradient becomes sufficiently small or if no lower function value can be found.

6.2.4 Steepest Descent Method

The simplest choice, which is known as the method of gradient descent or steepest descent[3] is to go in the direction of the negative gradient

$$\mathbf{s}_r = -\mathbf{g}_r \tag{6.82}$$

and to determine the step length by minimizing h along this direction

$$h(\lambda) = h(\mathbf{x}_r - \lambda \mathbf{g}_r) = \min. \tag{6.83}$$

Obviously two consecutive steps are orthogonal to each other since

$$0 = \frac{\partial}{\partial \lambda} h(\mathbf{x}_{r+1} - \lambda \mathbf{g}_r)_{|\lambda=0} = -\mathbf{g}_{r+1}^T \mathbf{g}_r. \tag{6.84}$$

[3]Which should not be confused with the method of steepest descent for the approximate calculation of integrals.

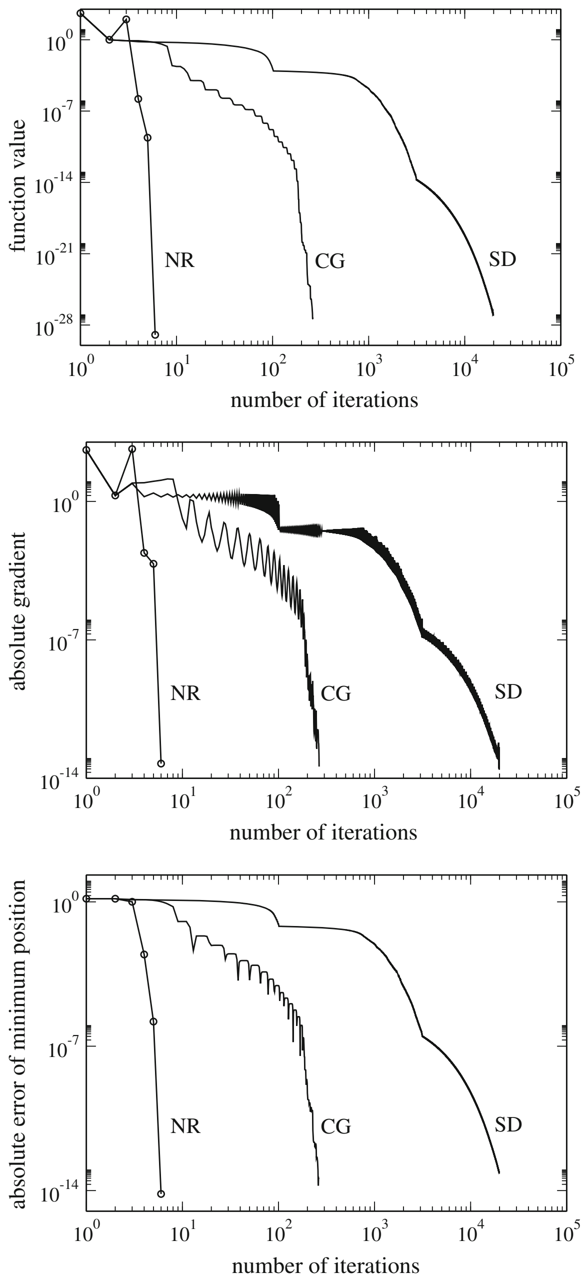


Fig. 6.22 (Function minimization) The minimum of the Rosenbrock function $h(x, y) = 100(y - x^2)^2 + (1 - x)^2$ is determined with different methods. Conjugate (CG) gradients converge much faster than steepest descent (SD). Starting at $(x, y) = (0, 2)$, Newton–Raphson (NR) reaches the minimum at $x = y = 1$ within only 5 iterations to machine precision

This can lead to a zig-zagging behavior and a very slow convergence of this method (Figs. 6.22 and 6.23).

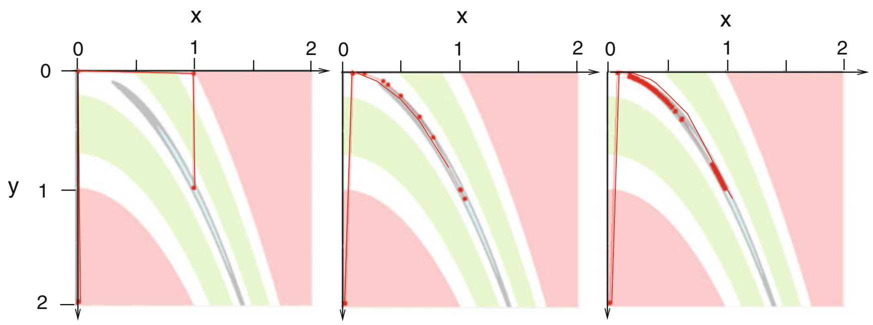


Fig. 6.23 (Minimization of the Rosenbrock function) ***Left*** Newton–Raphson finds the minimum after 5 steps within machine precision. ***Middle*** conjugate gradients reduce the gradient to 4×10^{-14} after 265 steps. ***Right*** The steepest descent method needs 20000 steps to reduce the gradient to 5×10^{-14}. *Red lines* show the minimization pathway. *Colored areas* indicate the function value (*light blue* < 0.1, *grey* $0.1 \ldots 0.5$, *green* $5 \ldots 50$, *pink* > 100). Screen shots taken from problem 6.2

6.2.5 Conjugate Gradient Method

This method is similar to the steepest descent method but the search direction is iterated according to

$$\mathbf{s}_0 = -\mathbf{g}_0 \tag{6.85}$$

$$\mathbf{x}_{r+1} = \mathbf{x}_r + \lambda_r \mathbf{s}_r \tag{6.86}$$

$$\mathbf{s}_{r+1} = -\mathbf{g}_{r+1} + \beta_{r+1}\mathbf{s}_r \tag{6.87}$$

where λ_r is chosen to minimize $h(\mathbf{x}_{r+1})$ and the simplest choice for β is made by Fletcher and Rieves [65]

$$\beta_{r+1} = \frac{g_{r+1}^2}{g_r^2}. \tag{6.88}$$

This method was devised to minimize a quadratic function and to solve the related system of linear equations, but it is also very efficient for more complicated functions (Sect. 5.6.4).

6.2.6 Newton–Raphson Method

The first order Newton–Raphson method uses the iteration

$$\mathbf{x}_{r+1} = \mathbf{x}_r - H(\mathbf{x}_r)^{-1}\mathbf{g}(\mathbf{x}_r). \tag{6.89}$$

The search direction is

$$\mathbf{s} = H^{-1}\mathbf{g} \tag{6.90}$$

and the step length is $\lambda = 1$. This method converges fast if the starting point is close to the minimum. However, calculation of the Hessian can be very time consuming (Fig. 6.22).

6.2.7 Quasi-Newton Methods

Calculation of the full Hessian matrix as needed for the Newton–Raphson method can be very time consuming. Quasi-Newton methods use instead an approximation to the Hessian which is updated during each iteration. From the Taylor series

$$h(\mathbf{x}) = h_0 + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \cdots \tag{6.91}$$

we obtain the gradient

$$\mathbf{g}(\mathbf{x}_r) = \mathbf{b} + H\mathbf{x}_r + \cdots = \mathbf{g}(\mathbf{x}_{r-1}) + H(\mathbf{x}_r - \mathbf{x}_{r-1}) + \cdots . \tag{6.92}$$

Defining the differences

$$\mathbf{d}_r = \mathbf{x}_{r+1} - \mathbf{x}_r \tag{6.93}$$

$$\mathbf{y}_r = \mathbf{g}_{r+1} - \mathbf{g}_r \tag{6.94}$$

and neglecting higher order terms we obtain the quasi-Newton or secant condition

$$H\mathbf{d}_r = \mathbf{y}_r. \tag{6.95}$$

We want to approximate the true Hessian by a series of matrices H_r which are updated during each iteration to sum up all the information gathered so far. Since the Hessian is symmetric and positive definite, this also has to be demanded for the H_r.[4] This cannot be achieved by a rank one update matrix. Popular methods use a symmetric rank two update of the form

$$H_{r+1} = H_r + \alpha\mathbf{u}\mathbf{u}^T + \beta\mathbf{v}\mathbf{v}^T. \tag{6.96}$$

The Quasi-Newton condition then gives

$$H_{r+1}\mathbf{d}_r = H_r\mathbf{d}_r + \alpha(\mathbf{u}^T\mathbf{d}_r)\mathbf{u} + \beta(\mathbf{v}^T\mathbf{d}_r)\mathbf{v} = \mathbf{y}_r \tag{6.97}$$

[4]This is a major difference to the Quasi Newton methods for root finding (6.1.9).

hence $H_r\mathbf{d}_r - \mathbf{y}_r$ must be a linear combination of $\mathbf{u}$ and $\mathbf{v}$. Making the simple choice

$$\mathbf{u} = \mathbf{y}_r \quad \mathbf{v} = H_r\mathbf{d}_r \tag{6.98}$$

and assuming that these two vectors are linearly independent, we find

$$\beta = -\frac{1}{(\mathbf{v}^T\mathbf{d}_r)} \quad = -\frac{1}{(\mathbf{d}_r^T H_r \mathbf{d}_r)} \tag{6.99}$$

$$\alpha = \frac{1}{(\mathbf{u}^T\mathbf{d}_r)} = \frac{1}{(\mathbf{y}_r^T\mathbf{d}_r)} \tag{6.100}$$

which together defines the very popular BFGS (Broyden, Fletcher, Goldfarb, Shanno) method [66–69]

$$H_{r+1} = H_r + \frac{\mathbf{y_r}\mathbf{y}_r^T}{\mathbf{y}_r^T\mathbf{d}_r} - \frac{(H_r\mathbf{d}_r)(H_r\mathbf{d}_r)^T}{\mathbf{d}_r^T H_r\mathbf{d}_r}. \tag{6.101}$$

Alternatively the DFP method by Davidon, Fletcher and Powell, directly updates the inverse Hessian matrix $B = H^{-1}$ according to

$$B_{r+1} = B_r + \frac{\mathbf{d}_r\mathbf{d}_r^T}{\mathbf{y}_r^T\mathbf{d}_r} - \frac{(B_r\mathbf{y}_r)(B_r\mathbf{y}_r)^T}{\mathbf{y}_r^T B_r\mathbf{y}_r}. \tag{6.102}$$

Both of these methods can be inverted with the help of the Sherman–Morrison formula to give

$$B_{r+1} = B_r + \frac{(\mathbf{d}_r - B_r\mathbf{y}_r)\mathbf{d}_r^T + \mathbf{d}_r(\mathbf{d}_r - B\mathbf{y}_r)^T}{\mathbf{y}_r^T\mathbf{d}_r} - \frac{(\mathbf{d}_r - B_r\mathbf{y}_r)^T\mathbf{y}}{(\mathbf{y}_r^T\mathbf{d})^2}\mathbf{d}\mathbf{d}^T \tag{6.103}$$

$$H_{r+1} = H_r + \frac{(\mathbf{y}_r - H_r\mathbf{d}_r)\mathbf{y}_r^T + \mathbf{y}_r(\mathbf{y}_r - H_r\mathbf{d}_r)^T}{\mathbf{y}_r^T\mathbf{d}_r} - \frac{(\mathbf{y}_r - H_r\mathbf{d}_r)\mathbf{d}_r}{(\mathbf{y}_r^T\mathbf{d}_r)^2}\mathbf{y}_r\mathbf{y}_r^T. \tag{6.104}$$

Problems

Problem 6.1 Root Finding Methods

This computer experiment searches roots of several test functions:

$f(x) = x^n - 2 \quad n = 1, 2, 3, 4$ (Fig. 6.10)
$f(x) = 5\sin(5x)$
$f(x) = (\cos(2x))^2 - x^2$
$f(x) = 5\left(\sqrt{|x+2|} - 1\right)$
$f(x) = \mathrm{e}^{-x}\ln x$

$f(x) = (x-1)^3$ (Fig. 6.11)
$f(x) = x^{25}$ (Fig. 6.12)

You can vary the initial interval or starting value and compare the behavior of different methods:

- bisection
- regula falsi
- Dekker's method
- Brent's method
- Chandrupatla's method
- Newton–Raphson method

Problem 6.2 Stationary Points

This computer experiment searches a local minimum of the Rosenbrock function[5]

$$h(x, y) = 100(y - x^2)^2 + (1-x)^2. \tag{6.105}$$

- The method of steepest descent minimizes $h(x, y)$ along the search direction

$$s_x^{(n)} = -g_x^{(n)} = -400x(x_n^2 - y_n) - 2(x_n - 1) \tag{6.106}$$
$$s_y^{(n)} = -g_y^{(n)} = -200(y_n - x_n^2). \tag{6.107}$$

- Conjugate gradients make use of the search direction

$$s_x^{(n)} = -g_x^{(n)} + \beta_n s_x^{(n-1)} \tag{6.108}$$
$$s_y^{(n)} = -g_y^{(n)} + \beta_n s_y^{(n-1)}. \tag{6.109}$$

- The Newton–Raphson method needs the inverse Hessian

$$H^{-1} = \frac{1}{\det(H)} \begin{pmatrix} h_{yy} & -h_{xy} \\ -h_{xy} & h_{xx} \end{pmatrix} \tag{6.110}$$

$$\det(H) = h_{xx}h_{yy} - h_{xy}^2 \tag{6.111}$$
$$h_{xx} = 1200x^2 - 400y + 2 \quad h_{yy} = 200 \quad h_{xy} = -400x \tag{6.112}$$

and iterates according to

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} - H^{-1} \begin{pmatrix} g_x^n \\ q_y^n \end{pmatrix}. \tag{6.113}$$

You can choose an initial point (x_0, y_0). The iteration stops if the gradient norm falls below 10^{-14} or if the line search fails to find a lower function value.

[5]A well known test function for minimization algorithms.