one-dimensional sub-minimization. Turn to §10.6 for detailed discussion and implementation.

- The second family goes under the names quasi-Newton or variable metric methods, as typified by the Davidon-Fletcher-Powell (DFP) algorithm (sometimes referred to just as Fletcher-Powell) or the closely related Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. These methods require of order $N^2$ storage, require derivative calculations and one-dimensional sub-minimization. Details are in §10.7.

You are now ready to proceed with scaling the peaks (and/or plumbing the depths) of practical optimization.

CITED REFERENCES AND FURTHER READING:

10.1 Golden Section Search in One Dimension

Recall how the bisection method finds roots of functions in one dimension (§9.1): The root is supposed to have been bracketed in an interval $(a, b)$. One then evaluates the function at an intermediate point $x$ and obtains a new, smaller bracketing interval, either $(a, x)$ or $(x, b)$. The process continues until the bracketing interval is acceptably small. It is optimal to choose $x$ to be the midpoint of $(a, b)$ so that the decrease in the interval length is maximized when the function is as uncooperative as it can be, i.e., when the luck of the draw forces you to take the bigger bisected segment.

There is a precise, though slightly subtle, translation of these considerations to the minimization problem: What does it mean to bracket a minimum? A root of a function is known to be bracketed by a pair of points, $a$ and $b$, when the function has opposite sign at those two points. A minimum, by contrast, is known to be bracketed only when there is a triplet of points, $a < b < c$ (or $c < b < a$), such that $f(b)$ is less than both $f(a)$ and $f(c)$. In this case we know that the function (if it is nonsingular) has a minimum in the interval $(a, c)$.

The analog of bisection is to choose a new point $x$, either between $a$ and $b$ or between $b$ and $c$. Suppose, to be specific, that we make the latter choice. Then we evaluate $f(x)$. If $f(b) < f(x)$, then the new bracketing triplet of points is $(a, b, x)$;
Figure 10.1.1. Successive bracketing of a minimum. The minimum is originally bracketed by points 1,3,2. The function is evaluated at 4, which replaces 2; then at 5, which replaces 1; then at 6, which replaces 4. The rule at each stage is to keep a center point that is lower than the two outside points. After the steps shown, the minimum is bracketed by points 5,3,6.

contrariwise, if \( f(b) > f(x) \), then the new bracketing triplet is \((b, x, c)\). In all cases the middle point of the new triplet is the abscissa whose ordinate is the best minimum achieved so far; see Figure 10.1.1. We continue the process of bracketing until the distance between the two outer points of the triplet is tolerably small.

How small is “tolerably” small? For a minimum located at a value \( b \), you might naively think that you will be able to bracket it in as small a range as \((1 - \epsilon)b < b < (1 + \epsilon)b\), where \( \epsilon \) is your computer’s floating-point precision, a number like \(3 \times 10^{-8}\) (single precision) or \(10^{-15}\) (double precision). Not so! In general, the shape of your function \( f(x) \) near \( b \) will be given by Taylor’s theorem

\[
f(x) \approx f(b) + \frac{1}{2} f''(b)(x - b)^2 \tag{10.1.1}
\]

The second term will be negligible compared to the first (that is, will be a factor \( \epsilon \) smaller and will act just like zero when added to it) whenever

\[
|x - b| < \sqrt{\epsilon}|b| \sqrt{\frac{2|f(b)|}{b^2f''(b)}} \tag{10.1.2}
\]

The reason for writing the right-hand side in this way is that, for most functions, the final square root is a number of order unity. Therefore, as a rule of thumb, it is hopeless to ask for a bracketing interval of width less than \( \sqrt{\epsilon} \) times its central value, a fractional width of only about \(10^{-4}\) (single precision) or \(3 \times 10^{-8}\) (double precision). Knowing this inescapable fact will save you a lot of useless bisections!

The minimum-finding routines of this chapter will often call for a user-supplied argument \( tol \), and return with an abscissa whose fractional precision is about \( \pm tol \) (bracketing interval of fractional size about \(2 \times tol\)). Unless you have a better
estimate for the right-hand side of equation (10.1.2), you should set \( \text{tol} \) equal to (not much less than) the square root of your machine’s floating-point precision, since smaller values will gain you nothing.

It remains to decide on a strategy for choosing the new point \( x \), given \((a, b, c)\). Suppose that \( b \) is a fraction \( w \) of the way between \( a \) and \( c \), \[ \frac{b - a}{c - a} = w \quad \frac{c - b}{c - a} = 1 - w \] (10.1.3)

Also suppose that our next trial point \( x \) is an additional fraction \( z \) beyond \( b \), \[ \frac{x - b}{c - a} = z \] (10.1.4)

Then the next bracketing segment will either be of length \( w + z \) relative to the current one, or else of length \( 1 - w \). If we want to minimize the worst case possibility, then we will choose \( z \) to make these equal, namely \[ z = 1 - 2w \] (10.1.5)

We see at once that the new point is the symmetric point to \( b \) in the original interval, namely with \(|b - a| = |x - c|\). This implies that the point \( x \) lies in the larger of the two segments (\( z \) is positive only if \( w < 1/2 \)).

But where in the larger segment? Where did the value of \( w \) itself come from? Presumably from the previous stage of applying our same strategy. Therefore, if \( z \) is chosen to be optimal, then so was \( w \) before it. This scale similarity implies that \( x \) should be the same fraction of the way from \( b \) to \( c \) (if that is the bigger segment) as was \( b \) from \( a \) to \( c \), in other words, \[ \frac{z}{1 - w} = w \] (10.1.6)

Equations (10.1.5) and (10.1.6) give the quadratic equation

\[ w^2 - 3w + 1 = 0 \quad \text{yielding} \quad w = \frac{3 - \sqrt{5}}{2} \approx 0.38197 \] (10.1.7)

In other words, the optimal bracketing interval \((a, b, c)\) has its middle point \( b \) a fractional distance 0.38197 from one end (say, \( a \)), and 0.61803 from the other end (say, \( b \)). These fractions are those of the so-called golden mean or golden section, whose supposedly aesthetic properties hark back to the ancient Pythagoreans. This optimal method of function minimization, the analog of the bisection method for finding zeros, is thus called the golden section search, summarized as follows:

Given, at each stage, a bracketing triplet of points, the next point to be tried is that which is a fraction 0.38197 into the larger of the two intervals (measuring from the central point of the triplet). If you start out with a bracketing triplet whose segments are not in the golden ratios, the procedure of choosing successive points at the golden mean point of the larger segment will quickly converge you to the proper, self-replicating ratios.

The golden section search guarantees that each new function evaluation will (after self-replicating ratios have been achieved) bracket the minimum to an interval...
just 0.61803 times the size of the preceding interval. This is comparable to, but not quite as good as, the 0.50000 that holds when finding roots by bisection. Note that the convergence is linear (in the language of Chapter 9), meaning that successive significant figures are won linearly with additional function evaluations. In the next section we will give a superlinear method, where the rate at which successive significant figures are liberated increases with each successive function evaluation.

**Routine for Initially Bracketing a Minimum**

The preceding discussion has assumed that you are able to bracket the minimum in the first place. We consider this initial bracketing to be an essential part of any one-dimensional minimization. There are some one-dimensional algorithms that do not require a rigorous initial bracketing. However, we would never trade the secure feeling of knowing that a minimum is “in there somewhere” for the dubious reduction of function evaluations that these nonbracketing routines may promise. Please bracket your minima (or, for that matter, your zeros) before isolating them!

There is not much theory as to how to do this bracketing. Obviously you want to step downhill. But how far? We like to take larger and larger steps, starting with some (wild?) initial guess and then increasing the stepsize at each step either by a constant factor, or else by the result of a parabolic extrapolation of the preceding points that is designed to take us to the extrapolated turning point. It doesn’t much matter if the steps get big. After all, we are stepping downhill, so we already have the left and middle points of the bracketing triplet. We just need to take a big enough step to stop the downhill trend and get a high third point.

Our standard routine is this:

```fortran
SUBROUTINE mnbrak(ax,bx,cx,fa,fb,fc,func)
REAL ax,bx,cx,fa,fb,fc,func,GOLD,GLIMIT,TINY
EXTERNAL func
PARAMETER (GOLD=1.618034, GLIMIT=100., TINY=1.e-20)
Given a function func, and given distinct initial points ax and bx, this routine searches
in the downhill direction (defined by the function as evaluated at the initial points) and
returns new points ax, bx, cx that bracket a minimum of the function. Also returned are
the function values at the three points, fa, fb, fc.
Parameters: GOLD is the default ratio by which successive intervals are magnified; GLIMIT
is the maximum magnification allowed for a parabolic-fit step.
REAL dum,fu,q,r,u,ulim
fa=func(ax)
fb=func(bx)
if(fb.gt.fa)then
  Switch roles of a and b so that we can go downhill in the
direction from a to b.
  dum=ax
  ax=bx
  bx=dum
  dum=fb
  fb=fa
  fa=dum
endif

1 cx=bx+GOLD*(bx-ax)
fu=func(cx)
if(fb.ge.fc)then
  "do while": keep returning here until we bracket.
  r=(bx-ax)*(fb-fc)
  q=(bx-cx)*(fb-fa)
  u=bx-((bx-cx)*q-(bx-ax)*r)/(2.*sign(max(abs(q-r),TINY),q-r))
  ulim=bx+GLIMIT*(cx-bx)
  We won’t go farther than this. Test various possibilities:
  if((bx-u)*(u-cx).gt.0.)then
    Parabolic u is between b and c: try it.
    fu=func(u)
  endif
```

Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software. Permission is granted for internet use users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcanfirm@cambridge.org (outside North America).
if(fu.lt.fc) then
  Got a minimum between b and c.
  ax=bx
  fa=fb
  bx=u
  fb=fu
  return
else if(fu.gt.fb) then
  Got a minimum between a and u.
  cx=u
  fc=fu
  return
endif

u=cx+GOLD*(cx-bx)
fu=func(u)
else if((cx-u)*(u-ulim).gt.0.) then
  Parabolic fit was no use. Use default magnification.
  fu=func(u)
endif

else if((cx-u)*(u-ulim).ge.0.) then
  Parabolic fit is between c and its allowed limit.
  fu=func(u)
endif

else
  Limit parabolic u to maximum allowed value.
  u=ulim
  fu=func(u)
else
  Reject parabolic u, use default magnification.
  u=cx+GOLD*(cx-bx)
  fu=func(u)
endif

ax=bx
Eliminate oldest point and continue.
bx=cx
cx=u
fa=fb
fb=fc
fc=fu
goto 1
endif
return
END

(Because of the housekeeping involved in moving around three or four points and their function values, the above program ends up looking deceptively formidable. That is true of several other programs in this chapter as well. The underlying ideas, however, are quite simple.)

Routine for Golden Section Search

FUNCTION golden(ax,bx,cx,f,tol,xmin)
REAL golden,ax,bx,cx,tol,xmin,f,R,C
EXTERNAL f
PARAMETER (R=.61803399,C=1.-R)

Given a function f, and given a bracketing triplet of abscissas ax, bx, cx (such that bx is between ax and cx, and f(bx) is less than both f(ax) and f(cx)), this routine performs a golden section search for the minimum, isolating it to a fractional precision of about tol. The abscissa of the minimum is returned as xmin, and the minimum function value is returned as golden, the returned function value.

Parameters: The golden ratios.

REAL f1,f2,x0,x1,x2,x3
x0=ax

At any given time we will keep track of four points, x0,x1,x2,x3.
10.2 Parabolic Interpolation and Brent’s Method

in One Dimension

We already tipped our hand about the desirability of parabolic interpolation in the previous section’s mnbrak routine, but it is now time to be more explicit. A golden section search is designed to handle, in effect, the worst possible case of function minimization, with the uncooperative minimum hunted down and cornered like a scared rabbit. But why assume the worst? If the function is nicely parabolic near to the minimum — surely the generic case for sufficiently smooth functions — then the parabola fitted through any three points ought to take us in a single leap to the minimum, or at least very near to it (see Figure 10.2.1). Since we want to find an abscissa rather than an ordinate, the procedure is technically called inverse parabolic interpolation.

The formula for the abscissa $x$ that is the minimum of a parabola through three points $f(a)$, $f(b)$, and $f(c)$ is

$$
x = b - \frac{1}{2} \frac{(b-a)^2 [f(b) - f(c)] - (b-c)^2 [f(b) - f(a)]}{(b-a) [f(b) - f(c)] - (b-c) [f(b) - f(a)]}
$$

(10.2.1)