the root lies near 10^{26}. One might thus think to specify convergence by a relative (fractional) criterion, but this becomes unworkable for roots near zero. To be most general, the routines below will require you to specify an absolute tolerance, such that iterations continue until the interval becomes smaller than this tolerance in absolute units. Usually you may wish to take the tolerance to be $\epsilon(|x_1|+|x_2|)/2$ where ϵ is the machine precision and x_1 and x_2 are the initial brackets. When the root lies near zero you ought to consider carefully what reasonable tolerance means for your function. The following routine quits after 40 bisections in any event, with $2^{-40} \approx 10^{-12}$.

```
FUNCTION rtbis(func,x1,x2,xacc)
INTEGER JMAX
REAL rtbis,x1,x2,xacc,func
EXTERNAL func
PARAMETER (JMAX=40)          Maximum allowed number of bisections.
   Using bisection, find the root of a function func known to lie between x1 and x2. The
   root, returned as rtbis, will be refined until its accuracy is ±xacc.
INTEGER j
REAL dx,f,fmid,xmid
fmid=func(x2)
f=func(x1)
if(f*fmid.ge.0.) pause 'root must be bracketed in rtbis'
if(f.lt.0.)then              Orient the search so that f>0 lies at x+dx.
    rtbis=x1
    dx=x2-x1
else
    rtbis=x2
    dx=x1-x2
endif
do 11 j=1,JMAX               Bisection loop.
    dx=dx*.5
    xmid=rtbis+dx
    fmid=func(xmid)
    if(fmid.le.0.)rtbis=xmid
    if(abs(dx).lt.xacc .or. fmid.eq.0.) return
enddo 11
pause 'too many bisections in rtbis'
END
```

9.2 Secant Method, False Position Method, and Ridders' Method

For functions that are smooth near a root, the methods known respectively as *false position* (or *regula falsi*) and *secant method* generally converge faster than bisection. In both of these methods the function is assumed to be approximately linear in the local region of interest, and the next improvement in the root is taken as the point where the approximating line crosses the axis. After each iteration one of the previous boundary points is discarded in favor of the latest estimate of the root.

The *only* difference between the methods is that secant retains the most recent of the prior estimates (Figure 9.2.1; this requires an arbitrary choice on the first iteration), while false position retains that prior estimate for which the function value

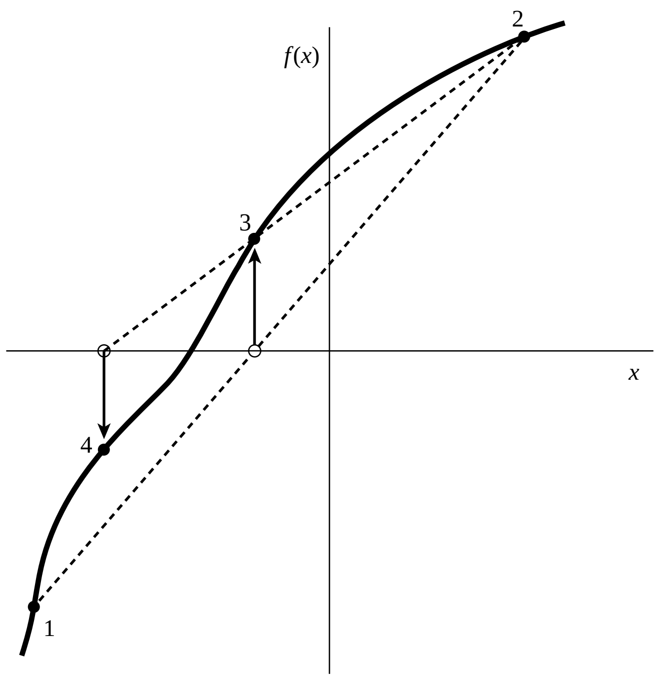

Figure 9.2.1. Secant method. Extrapolation or interpolation lines (dashed) are drawn through the two most recently evaluated points, whether or not they bracket the function. The points are numbered in the order that they are used.

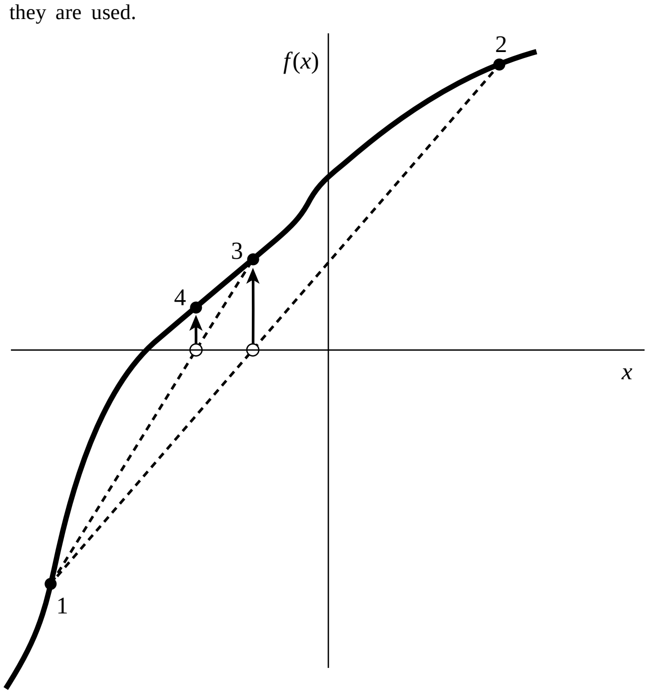

Figure 9.2.2. False position method. Interpolation lines (dashed) are drawn through the most recent points *that bracket the root*. In this example, point 1 thus remains "active" for many steps. False position converges less rapidly than the secant method, but it is more certain.

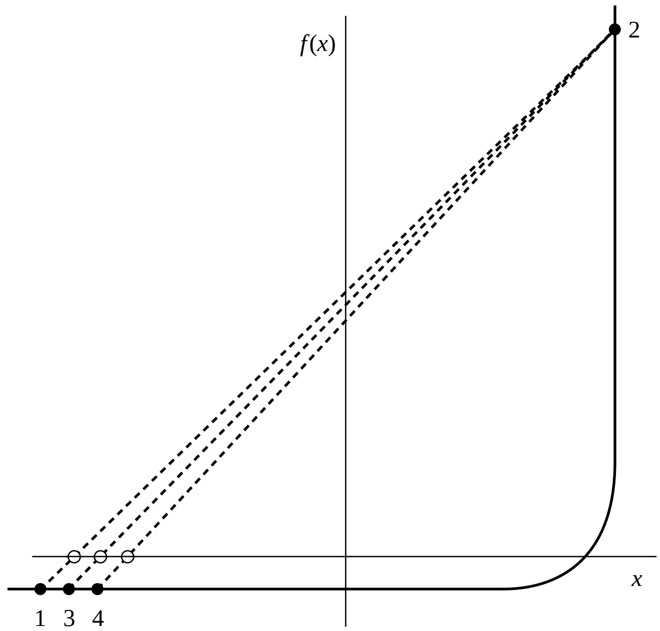

Figure 9.2.3. Example where both the secant and false position methods will take many iterations to arrive at the true root. This function would be difficult for many other root-finding methods.

has opposite sign from the function value at the current best estimate of the root, so that the two points continue to bracket the root (Figure 9.2.2). Mathematically, the secant method converges more rapidly near a root of a sufficiently continuous function. Its order of convergence can be shown to be the "golden ratio" $1.618\ldots$, so that

$$\lim_{k\to\infty} |\epsilon_{k+1}| \approx \text{const} \times |\epsilon_k|^{1.618} \tag{9.2.1}$$

The secant method has, however, the disadvantage that the root does not necessarily remain bracketed. For functions that are *not* sufficiently continuous, the algorithm can therefore not be guaranteed to converge: Local behavior might send it off towards infinity.

False position, since it sometimes keeps an older rather than newer function evaluation, has a lower order of convergence. Since the newer function value will *sometimes* be kept, the method is often superlinear, but estimation of its exact order is not so easy.

Here are sample implementations of these two related methods. While these methods are standard textbook fare, *Ridders' method*, described below, or *Brent's method*, in the next section, are almost always better choices. Figure 9.2.3 shows the behavior of secant and false-position methods in a difficult situation.

```
FUNCTION rtflsp(func,x1,x2,xacc)
INTEGER MAXIT
REAL rtflsp,x1,x2,xacc,func
EXTERNAL func
PARAMETER (MAXIT=30)          Set to the maximum allowed number of iterations.
```

```
    Using the false position method, find the root of a function func known to lie between x1
    and x2. The root, returned as rtflsp, is refined until its accuracy is ±xacc.
INTEGER j
REAL del,dx,f,fh,fl,swap,xh,xl
fl=func(x1)
fh=func(x2)                  Be sure the interval brackets a root.
if(fl*fh.gt.0.) pause 'root must be bracketed in rtflsp'
if(fl.lt.0.)then             Identify the limits so that xl corresponds to the low side.
    xl=x1
    xh=x2
else
    xl=x2
    xh=x1
    swap=fl
    fl=fh
    fh=swap
endif
dx=xh-xl
do 11 j=1,MAXIT              False position loop.
    rtflsp=xl+dx*fl/(fl-fh)  Increment with respect to latest value.
    f=func(rtflsp)
    if(f.lt.0.) then         Replace appropriate limit.
        del=xl-rtflsp
        xl=rtflsp
        fl=f
    else
        del=xh-rtflsp
        xh=rtflsp
        fh=f
    endif
    dx=xh-xl
    if(abs(del).lt.xacc.or.f.eq.0.)return        Convergence.
enddo 11
pause 'rtflsp exceed maximum iterations'
END
```

```
FUNCTION rtsec(func,x1,x2,xacc)
INTEGER MAXIT
REAL rtsec,x1,x2,xacc,func
EXTERNAL func
PARAMETER (MAXIT=30)         Maximum allowed number of iterations.
    Using the secant method, find the root of a function func thought to lie between x1 and
    x2. The root, returned as rtsec, is refined until its accuracy is ±xacc.
INTEGER j
REAL dx,f,fl,swap,xl
fl=func(x1)
f=func(x2)
if(abs(fl).lt.abs(f))then    Pick the bound with the smaller function value as the most
    rtsec=x1                     recent guess.
    xl=x2
    swap=fl
    fl=f
    f=swap
else
    xl=x1
    rtsec=x2
endif
do 11 j=1,MAXIT              Secant loop.
    dx=(xl-rtsec)*f/(f-fl)   Increment with respect to latest value.
    xl=rtsec
    fl=f
    rtsec=rtsec+dx
```

```
    f=func(rtsec)
    if(abs(dx).lt.xacc.or.f.eq.0.)return      Convergence.
enddo 11
pause 'rtsec exceed maximum iterations'
END
```

Ridders' Method

A powerful variant on false position is due to Ridders [1]. When a root is bracketed between x_1 and x_2, Ridders' method first evaluates the function at the midpoint $x_3 = (x_1 + x_2)/2$. It then factors out that unique exponential function which turns the residual function into a straight line. Specifically, it solves for a factor e^Q that gives

$$f(x_1) - 2f(x_3)e^Q + f(x_2)e^{2Q} = 0 \tag{9.2.2}$$

This is a quadratic equation in e^Q, which can be solved to give

$$e^Q = \frac{f(x_3) + \text{sign}[f(x_2)]\sqrt{f(x_3)^2 - f(x_1)f(x_2)}}{f(x_2)} \tag{9.2.3}$$

Now the false position method is applied, not to the values $f(x_1), f(x_3), f(x_2)$, but to the values $f(x_1), f(x_3)e^Q, f(x_2)e^{2Q}$, yielding a new guess for the root, x_4. The overall updating formula (incorporating the solution 9.2.3) is

$$x_4 = x_3 + (x_3 - x_1)\frac{\text{sign}[f(x_1) - f(x_2)]f(x_3)}{\sqrt{f(x_3)^2 - f(x_1)f(x_2)}} \tag{9.2.4}$$

Equation (9.2.4) has some very nice properties. First, x_4 is guaranteed to lie in the interval (x_1, x_2), so the method never jumps out of its brackets. Second, the convergence of successive applications of equation (9.2.4) is *quadratic*, that is, $m = 2$ in equation (9.1.4). Since each application of (9.2.4) requires two function evaluations, the actual order of the method is $\sqrt{2}$, not 2; but this is still quite respectably superlinear: the number of significant digits in the answer approximately *doubles* with each two function evaluations. Third, taking out the function's "bend" via exponential (that is, ratio) factors, rather than via a polynomial technique (e.g., fitting a parabola), turns out to give an extraordinarily robust algorithm. In both reliability and speed, Ridders' method is generally competitive with the more highly developed and better established (but more complicated) method of Van Wijngaarden, Dekker, and Brent, which we next discuss.

```
      FUNCTION zriddr(func,x1,x2,xacc)
      INTEGER MAXIT
      REAL zriddr,x1,x2,xacc,func,UNUSED
      PARAMETER (MAXIT=60,UNUSED=-1.11E30)
      EXTERNAL func
C     USES func
         Using Ridders' method, return the root of a function func known to lie between x1 and
         x2. The root, returned as zriddr, will be refined to an approximate accuracy xacc.
      INTEGER j
      REAL fh,fl,fm,fnew,s,xh,xl,xm,xnew
```

```
fl=func(x1)
fh=func(x2)
if((fl.gt.0..and.fh.lt.0.).or.(fl.lt.0..and.fh.gt.0.))then
    xl=x1
    xh=x2
    zriddr=UNUSED                          Any highly unlikely value, to simplify logic
    do 11 j=1,MAXIT                            below.
        xm=0.5*(xl+xh)
        fm=func(xm)                        First of two function evaluations per it-
        s=sqrt(fm**2-fl*fh)                    eration.
        if(s.eq.0.)return
        xnew=xm+(xm-xl)*(sign(1.,fl-fh)*fm/s)  Updating formula.
        if (abs(xnew-zriddr).le.xacc) return
        zriddr=xnew
        fnew=func(zriddr)                  Second of two function evaluations per
        if (fnew.eq.0.) return                 iteration.
        if(sign(fm,fnew).ne.fm) then       Bookkeeping to keep the root bracketed
            xl=xm                              on next iteration.
            fl=fm
            xh=zriddr
            fh=fnew
        else if(sign(fl,fnew).ne.fl) then
            xh=zriddr
            fh=fnew
        else if(sign(fh,fnew).ne.fh) then
            xl=zriddr
            fl=fnew
        else
            pause 'never get here in zriddr'
        endif
        if(abs(xh-xl).le.xacc) return
    enddo 11
    pause 'zriddr exceed maximum iterations'
else if (fl.eq.0.) then
    zriddr=x1
else if (fh.eq.0.) then
    zriddr=x2
else
    pause 'root must be bracketed in zriddr'
endif
return
END
```

CITED REFERENCES AND FURTHER READING:

Ralston, A., and Rabinowitz, P. 1978, *A First Course in Numerical Analysis*, 2nd ed. (New York: McGraw-Hill), §8.3.

Ostrowski, A.M. 1966, *Solutions of Equations and Systems of Equations*, 2nd ed. (New York: Academic Press), Chapter 12.

Ridders, C.J.F. 1979, *IEEE Transactions on Circuits and Systems*, vol. CAS-26, pp. 979–980. [1]

9.3 Van Wijngaarden–Dekker–Brent Method

While secant and false position formally converge faster than bisection, one finds in practice pathological functions for which bisection converges more rapidly.