2.0 NUMERICAL TECHNIQUES FOR SINGLE VARIABLE OPTIMIZATION

2.1 Introduction

In the previous chapter we reviewed some of the analytic properties of extrema and gave necessary and sufficient conditions for local extrema for a function of a single variable. That is important to know and we will also extend it to multivariable functions later but the question now arises: how do we get a computer to find the extrema of our single variable function? Those necessary and sufficient conditions discussed previously required the derivatives of the function. What if the derivative is unknown? For example, our optimization problem may be a physical process in which we have only a vague idea of the functional dependence between the variables.

In this chapter we will look at numerical techniques which can be implemented on a computer. These methods can be classified by the type of functional information they use to find the extrema. For example techniques which make use of function values only are called zeroth order techniques. Those which use, in addition, first and second derivative information are called first and second order techniques. One of the oldest techniques is the Newton-Raphson technique which is really a root finding technique and we shall consider it next.

2.2 Newton-Raphson Technique

Since we know that the minimum of *f(x)* must lie at a stationary point we look for the roots of the equation $f'(x) = 0$. (Thus we will somehow need to evaluate $f'(x)$ to use this method.) geometrically the method is very simple. We start with an initial guess, $x^{(0)}$, and the first and second derivative values of the function at this guess, $f'(x^{(0)})$ and $f''(x^{(0)})$. The derivatives allow us to follow the tangent line to the x-axis and define a new, hopefully improved, x value $x^{(1)}$ from

which the cycle repeats itself. We stop when the value of the derivative at the current x value is as close to zero as we so desire. The method is depicted in Figure 2.1.

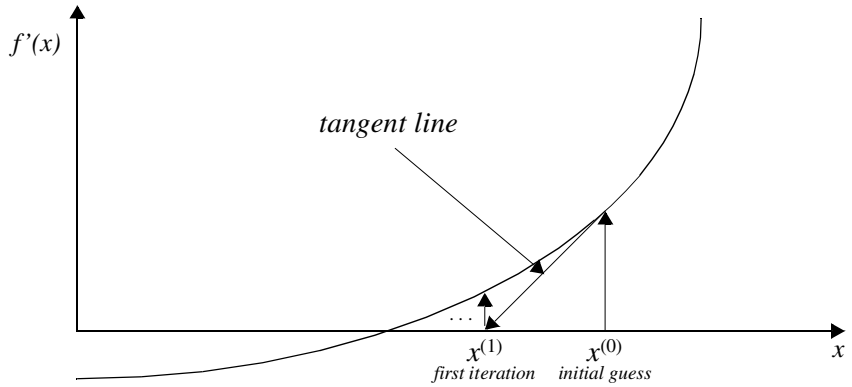

Figure 2.1 one iteration of the Newton-Raphson method

Algebraically we start with an initial guess $x^{(0)}$ and then use the fact that the slope of the tangent line at $x^{(0)}$ is given by

$$f''(x^{(0)}) = \frac{f'(x^{(0)}) - 0}{(x^{(0)} - x^{(1)})}$$

and use this to determine the next point, $x^{(1)}$ as

$$x^{(1)} = x^{(0)} - \frac{f'(x^{(0)})}{f''(x^{(0)})} \quad .$$

or in general, for the n^{th} iteration

$$x^{(n)} = x^{n-1} - \frac{f'(x^{(n-1)})}{f''(x^{(n-1)})} \tag{2.1}$$

The problem with this method is that it requires first and second derivatives and sometimes this scheme diverges depending not only on *f(x)* but on the initial guess.

2.2.1 Example - diverging iteration

Figure 2.2 depicts a few iterations of applying the Newton-Raphson technique in which the method diverges.

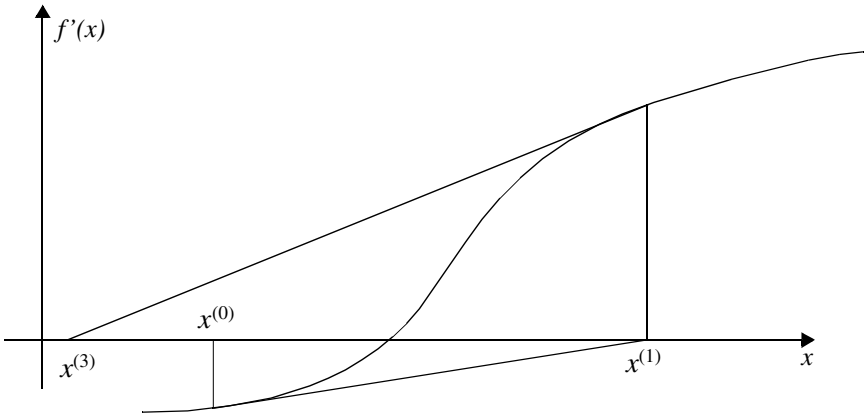

Figure 2.2 diverging iterations in Newton-Raphson method

2.2.2 **Example** - numerical

We now consider the following example:

$$\text{minimize } f(x) = -x^3 + \frac{3}{4}x^4$$

Since $f'(x) = -3x^2 + 3x^3$ and $f''(x) = -6x + 9x^2$ we form the following iteration:

$$x^{n+1} = x^n - \frac{3(x^n)^3 - 3(x^n)^2}{9(x^n)^2 - 6x^n} = x^n - \frac{(x^n)^2 - x^n}{3x^n - 2} = \frac{2(x^n)^2 - x^n}{3x^n - 2} \quad .$$

We can now try the iteration for different initial guesses.

initial guess: $x^{(0)} = 0$

$$(x^{(1)} = 0) \Rightarrow \text{stationary point}$$

initial guess: $x^{(0)} = 0.5$

$$(x^{(1)} = 0) \Rightarrow \text{stationary point}$$

initial guess: $x^{(0)} = .9\bar{}$

$$x^{(1)} = 1.029$$

$$x^{(2)} = 1.0015$$

$$(x^{(3)} = 1.0000045) \Rightarrow 1.0 = \text{stationary point}$$

initial guess: $x^{(0)} = 1.0$

$$x^{(1)} = 1.0$$

We can now evaluate *f(x)* at the two stationary points we've found:

$$f(0) = 0\ ,\ f(1) = -1/4$$

and since $f''(1) = 3 > 0 \Rightarrow$ local minimum therefore $x^* = 1 \Rightarrow$ local minimum. In fact you can convince yourself that it is the global minimum. (Note that the global maximum is at $x^* = \pm\infty$.)

2.3 Search Methods: Region Elimination

In general, we now concern ourselves with finding the minimum inside an "interval of uncertainty" or "bracketed interval". We evaluate our methods by comparing the number of function evaluations required to determine the solution with required accuracy. (Note: we always assume that our function is unimodal in bracketed interval.)

As can be seen from Figure 2.3 one function evaluation inside the bracketed interval [*a, b*] is not enough to reduce the interval of uncertainty. Both f_1 and f_2 have the same functional value at $x = c$ but f_2 has its minimum in the interval [*a, c*] whereas f_1 has its minimum in the interval [*c, b*].

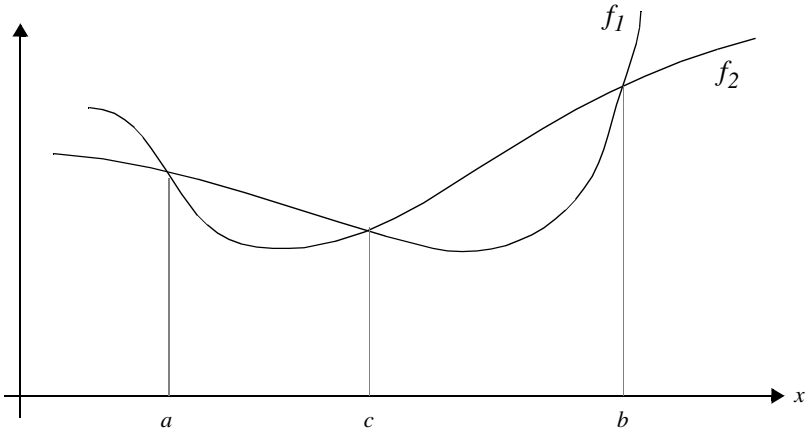

Figure 2.3 One function evaluation inside a bracketed interval

Thus with only one function evaluation at *c*, there are two possible functions and we cannot tell how to reduce the interval.

Now consider the same interval $I_0 = [a, b]$ with two function evaluations at $x = c$ and $x = d$ as shown in Figure 2.4. It is now clear that depending on the value of the function at $x = c$ as compared to at $x = d$ we can reduce the interval of uncertainty to either $I_1 = [c, b]$ or $I_1 = [a, d]$.

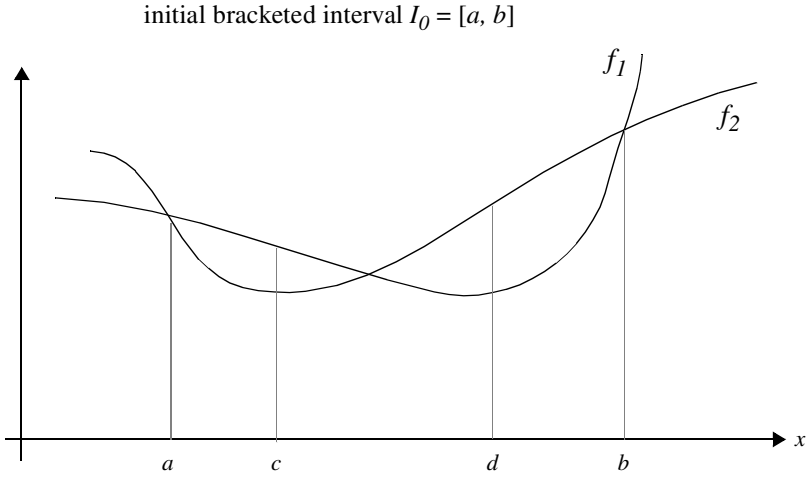

Figure 2.4 Two function evaluations inside bracketed interval

For the example shown in Figure 2.4 the new interval I_1 would be chosen as follows:

1) consider $f_1 : f_1(d) < f_1(c) \Rightarrow I_1 = [c,b]$,

2) consider $f_2 : f_2(d) > f_2(c) \Rightarrow I_1 = [a,d]$.

and in either case I_1 is a new, smaller interval in which the minimum must lie.

Theorem If $f(x)$ is *strictly* unimodal on $S = \{x \mid a \le x \le b\}$ (*i.e.* no plateau's), given two function evaluations, at c and d, $c < d$, *i.e.* $f(c)$, $f(d)$, then

a) if $f(c) < f(d) \Rightarrow x^* \in [a,d]$

b) if $f(c) > f(d) \Rightarrow x^* \in [c,b]$

c) if $f(c) = f(d) \Rightarrow x^* \in [c,d]$

Note: case (c) is not used in practice because numerically equality is hard to determine. ❐

The problem with non-strict unimodal functions (*i.e.* functions with *plateau's*) is that plateau's may give the now indeterminate condition. Therefore we usually assume that $f(x)$ is strictly unimodal.

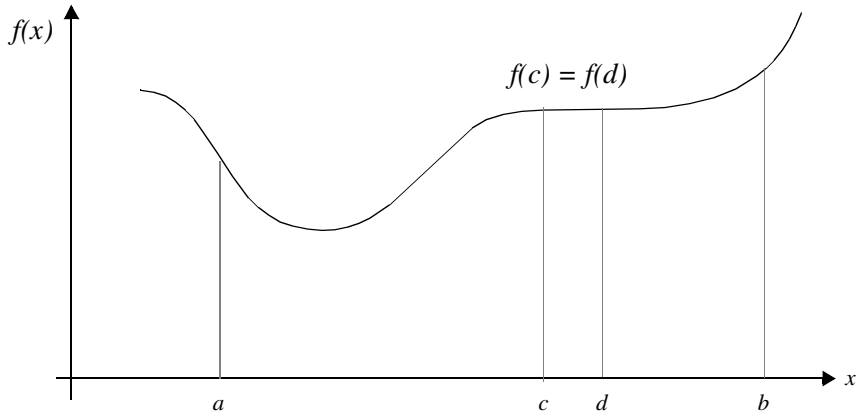

Figure 2.5 Function which is not strictly unimodal causes uncertainty in interval reduction

2.3.1 Interval Halving Search

The most obvious technique which arises from the previous discussion can be simply described as follows: given an initial interval $[a,b]$ which brackets a function $f(x)$ we sample at $1/4$, $1/2$ and $3/4$ of the interval as shown in Figure 2.6. The new interval will now become either $[a, x_m]$ if $f(x_L) < f(x_m)$ or $[x_m, b]$ if $f(x_u) < f(x_m)$. If neither of these is satisfied then the minimum must be between x_L and x_u.

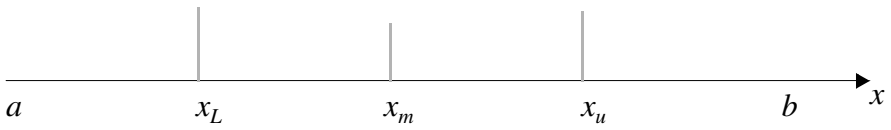

Figure 2.6 First step of interval halving search

The process continues with the new interval but now only two function evaluations will be required at 1/4 and 3/4 of the interval. The half way point is reused from the previous iteration. The process stops when the new interval is of size less than a user specified tolerance Δmin. An algorithm implementing this procedure is shown below.

We notice that after the initial interval reduction we require 2 function evaluations for each interval refinement of 1/2. Therefore after $2n+1$ function evaluation the interval is reduced by

$$I_{2n+1} = I_0(1/2)^n, \; n = 1, 2, 3\ldots \tag{2.2}$$

where I_o is the size of the initial interval.

Algorithm: Interval Halving Search

1. input a, b, Δmin // input the initial interval
2. set: $\Delta = b - a$, $x_m = a + \Delta/2$, $f_m = f(x_m)$
3. while $\Delta > \Delta min$ repeat
4. set: $x_L = a + \Delta/4$, $f_L = f(x_L)$
5. $x_u = b - \Delta/4$, $f_u = f(x_u)$
6. if $f_L < f_m$ then
7. set $b = x_m$, $x_m = x_L$, $f_m = f_L$, $\Delta = b - a$
8. else if $f_u < f_m$
9. set $a = x_m$, $x_m = x_u$, $f_m = f_u$, $\Delta = b - a$
10. else
11. set $a = x_L$, $b = x_u$, $\Delta = b - a$
12. end if
13. end if
14. end while
15. output [a, b] // final interval size is less than Δmin

2.3.2 Fibonacci Search Technique

We may now wonder if the interval halving technique is the best we can do. The following question comes to mind:

Question How should we pick N successive points in an interval to perform function evaluations such that the final reduced interval is the smallest possible irrespective of the properties of $f(x)$?

If N is the number of function evaluations then it is clear that for

$$N = 1 : \Rightarrow \text{cannot reduce interval - indeterminant}$$

for $N = 2$ it is clear that the best place to sample the function is at two points a δ distance apart at the center of our interval of uncertainty where δ is very small number. This is shown pictorially in Figure 2.7 where the lengths of the shaded vertical lines represent the value of the function at those points.

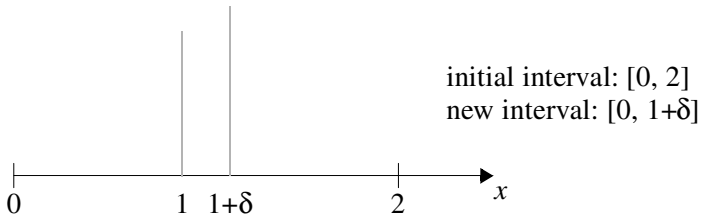

Figure 2.7 best location for two function evaluations

If the number of function evaluations we are allowed is $N = 3, 4, 5$ *etc*. then the best locations to sample the function given that all we know is that it is strictly unimodal are shown in Figure 2.8. Note that although we choose to use different initial intervals of uncertainty for each of the cases, this is not really a restriction since we can always normalize our interval to *any* size.

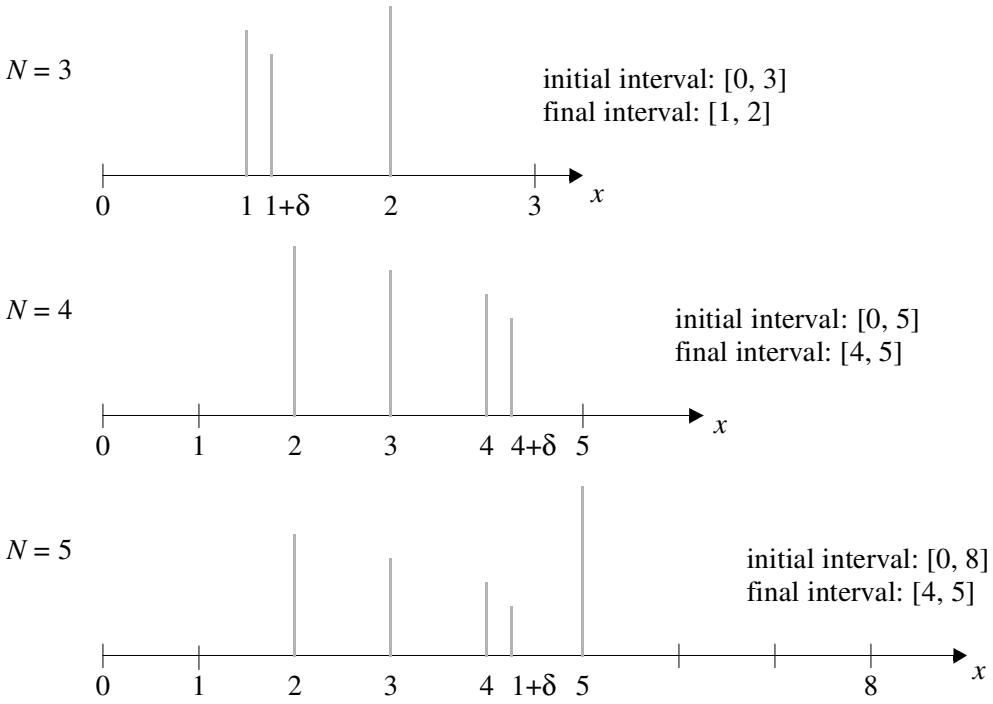

Figure 2.8 examples of optimal sampling locations

For the case $N = 3$, we start with an interval of size 3, $I_0 = [0, 3]$ and sample at $x = 1$ and $x = 2$. The result of this will be an interval of size 2 which for the case of Figure 2.8 is $I_1 = [0, 2]$. We know from Figure 2.7 that the best location to take our one remaining sample is

at the point $x = 1+\delta$. Notice that we already have the sample at $x = 1$ from the previous step. Our final interval is now of size 1 and is $I_2 = [1, 2]$.

The remaining two case shown in Figure 2.8 follow in a similar manner, each iteration consisting of only one new sample reduces our interval size because we are reusing a sample from the previous iteration.

It is not too difficult to see that the pattern of sample points used above is based on the *Fibonacci sequence* of numbers:

$$F_0 = F_1 = 1\,,\; F_i = F_{i-1} + F_{i-2}\,,\; i = 2(1)\infty\,.$$

Given an interval of proportion F_i and choosing two sample points at F_{i-2}, F_{i-1} .

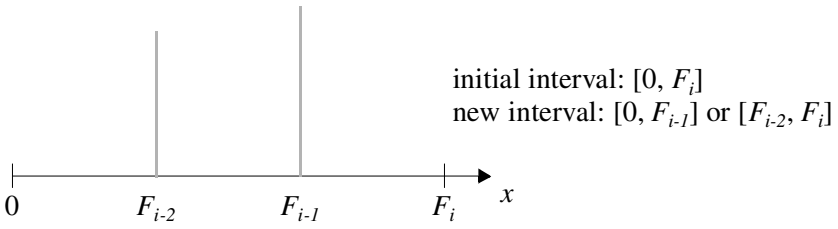

Figure 2.9 Fibonacci type sampling

The *new interval* will be either

$$[0, F_{i-1}] \text{ or } [F_{i-2}, F_i]$$

the first has size F_{i-1}, the second has size $F_i - F_{i-2} = F_{i-1}$ and therefore they are the same size. The next point is chosen as either F_{i-3} or $F_i - F_{i-3}$.

The interval reduction is equal to $I_N / I_0 = 1/F_N$ after N function evaluations. It has been shown that this is the best possible reduction for a given number of function evaluations. At each step the reduction is equal to F_i / F_{i+1}. The drawback of this procedure is that you must start with F_N and work backwards down the sequence.

It turns out that this technique is related to a simpler method called Golden Section search with

$$\tau = \lim_{i \to \infty} \frac{F_i}{F_{i+1}} = 0.618$$

called the golden section.

2.3.3 Golden Section Search

Given a normalized interval [0,1] where should two points x_1 and x_2 be chosen such that:

1) size of reduced interval is independent of function.
2) only one function evaluation per interval reduction.

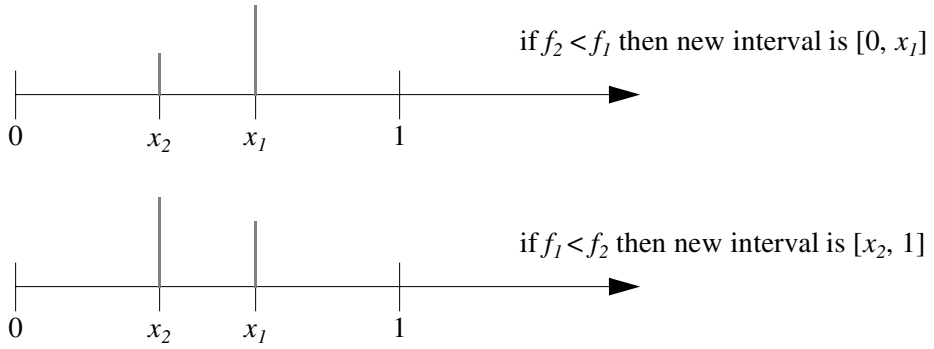

Figure 2.10 golden section interval reduction, initial interval is [0, 1].

Therefore by condition 1 above

$$x_1 - 0 = 1 - x_2 \text{ or } x_2 = 1 - x_1 \tag{2.3}$$

and by condition 2

$$\frac{x_1 - x_2}{x_1} = \frac{1 - x_1}{1} \tag{2.4}$$

$$x_1 - x_2 = x_1 - x_1^2 \Rightarrow x_1^2 = x_2 \tag{2.5}$$

from (2.3)

$$x_1^2 + x_1 - 1 = 0 \Rightarrow x_1 = \tau = .61803$$

$$x_2 = 1 - x_1 = .38197 = \theta = 1 - \tau = \tau^2$$

where we call τ the golden section and θ the golden mean.

After each function evaluation and comparison θ of the interval is eliminated or τ of the original interval remains. Thus after N function evaluations an interval of original size L will be reduced to a size

$$L\tau^{N-1}. \tag{2.6}$$

Algorithm: Golden Section search

1. input a_1, b_1, tol
2. set $c_1 = a_1 + (1-\tau)(b_1 - a_1), F_c = F(c_1)$
3. $d_1 = b_1 - (1-\tau)(b_1 - a_1), F_d = F(d_1)$
4. for $K = 1, 2, \ldots$ repeat
5. if $F_c < F_d$ then
6. set $a_{k+1} = a_k, b_{k+1} = d_k, d_{k+1} = c_k$
7. $c_{k+1} = a_{k+1} + (1-\tau)(b_{k+1} - a_{k+1})$
8. $F_d = F_c, F_c = F(c_{k+1})$
9. else
10. set $a_{k+1} = c_k, b_{k+1} = b_k, c_{k+1} = d_k$
11. $d_{k+1} = b_{k+1} - (1-\tau)(b_{k+1} - a_{k+1})$
12. $F_c = F_d, F_d = F(d_{k+1})$
13. end
14. end until $b_{k+1} - a_{k+1} < tol$

Notes: iterate until ‖interval‖ < tol

2.4 Polynomial Interpolation Methods

In an interval of uncertainty the function is approximated by a polynomial and the minimum of the polynomial is used to predict the minimum of the function.

2.4.1 Quadratic Interpolation

If $F(x) = px^2 + qx + r$ is the interpolating quadratic we need to find the coefficients *p*, *q*, and *r* given the end points of the interval [*a*,*b*] and *c* such that $a < c < b$ with $F_a = F(a)$ $F_b = F(b)$, $F_c = F(c)$. The coefficients satisfy the matrix equation

$$\begin{bmatrix} a^2 & a & 1 \\ b^2 & b & 1 \\ c^2 & c & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} F_a \\ F_b \\ F_c \end{bmatrix} \tag{2.7}$$

which can be easily solved analytically. Once we have the coefficients we need to find the minimum of *F(x)* and thus

$$\frac{d}{dx}F(x) = g(x^*) = 2px^* + q = 0$$

$$x^* = -\frac{q}{2p}$$

and therefore we only need the q and p coefficients. These can be solved using Cramer's rule:

$$p = \frac{1}{\Delta}\begin{vmatrix} F_a & a & 1 \\ F_b & b & 1 \\ F_c & c & 1 \end{vmatrix} = \frac{F_a(b-c) + F_b(c-a) + F_c(a-b)}{\Delta} \tag{2.8}$$

$$q = \frac{1}{\Delta}\begin{vmatrix} a^2 & F_a & 1 \\ b^2 & F_b & 1 \\ c^2 & F_c & 1 \end{vmatrix} = -\frac{F_a(b^2-c^2) + F_b(a^2-c^2) - F_c(a^2-b^2)}{\Delta} \tag{2.9}$$

and therefore

$$x^* = -\frac{q}{2p} = \frac{1}{2}\frac{F_a(b^2-c^2) + F_b(c^2-a^2) + F_c(a^2-b^2)}{F_a(b-c) + F_b(c-a) + F_c(a-b)}$$

Notice that x^* must lie in $[a,b]$ if $F_c < F_a$ and $F_c < F_b \Leftarrow$ bracketing interval. Also note that $f''(x^*) = 2p(x^*)$ and therefore if $p(x^*) > 0 \Rightarrow$ local minimum of quadratic whereas if $p(x^*) < 0 \Rightarrow$ local maximum.

We now evaluate $f(x^*)$ and compare it to $f(c)$ in order to reduce the interval of uncertainty.

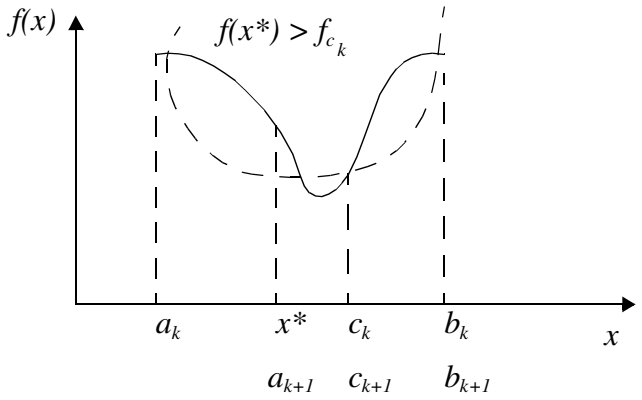

Figure 2.11 $f(x^*) > f_{c_k}$ $\Rightarrow$ new interval is $[a_{k+1}, b_{k+1}]$ and new c value is c_{k+1}

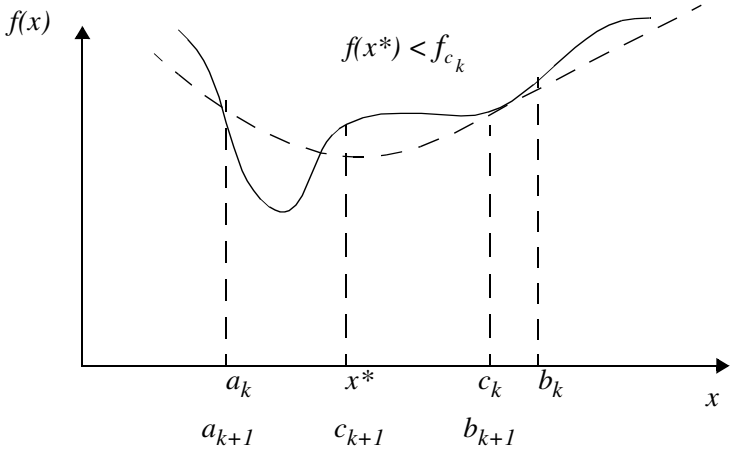

Figure 2.12 $f(x^*) < f_{c_k}$ $\Rightarrow$ new interval is $[a_{k+1}, b_{k+1}]$ and new c value is c_{k+1}

Algorithm: Quadratic interpolation search

1. input $a_1, b_1, c_1, xtol, ftol$
2. set $F_a = F(a_1), F_b = F(b_1), F_c = F(c_1)$
3. for $k = 1,2, \ldots$ repeat
4. set $x^* = \frac{1}{2}\frac{F_a(b_k^2 - c_k^2) + F_b(c_k^2 - a_k^2) + F_c(a_k^2 - b_k^2)}{F_a(b_k - c_k) + F_b(c_k - a_k) + F_c(a_k - b_k)}$
5. $F_x = F(x^*)$
6. if $x^* > c_k$ and $F_x < F_c$ then
7. set $a_{k+1} = c_k, b_{k+1} = b_k, c_{k+1} = x^*$
8. $F_a = F_c, F_c = F_x$
9. else if $x^* > c_k$ and $F_x > F_c$ then
10. set $a_{k+1} = a_k, b_{k+1} = x^*, c_{k+1} = c_k$
11. $F_b = F_x$
12. else if $x^* < c_k$ and $F_x > F_c$ then
13. set $a_{k+1} = x^*, b_{k+1} = b_k, c_{k+1} = c_k$
14. $F_a = F_x$
15. else
16. set $a_{k+1} = a_k, b_{k+1} = c_k, c_{k+1} = x^*$
17. $F_b = F_c, F_c = F_x$
18. end
19. end until $(b_{k+1} - a_{k+1}) < xtol$ or $(F(c_k) - F(c_{k+1}))/F(c_k) < ftol$

Notes: stop when interval reduced to *xtol* or when relative change in function value < *ftol*

2.4.2 Combined Quadratic Approximation and Golden Section : Brent's Method in 1-D

Only an outline will be given of this method. The details can be found in *Numerical Recipes*.

It is a hybrid method where an interval of uncertainty is maintained $[a_k,b_k]$ but the extreme points are not necessarily used to compute the quadratic.

Four additional points are used : $\mu_k, \upsilon_k, \omega_k, x_k$

x_k - the point with the lowest function value

ω_k - has the next lowest function value

υ_k - previous value of ω_k

μ_k - the latest point at which the function has been evaluated.

$F_\mu = F(\mu)$, $F_\upsilon = F(\upsilon)$, $F_\omega = F(w)$, $F_x = F(x)$

initially: a_1, b_1 are given and μ_1 is undefined

$\upsilon_1 = \omega_1 = x_1 = a_1 + (1 - \tau)(b_1 - a_1)$ (i. e. start with Golden section).

Parabolic fit is tried on F_υ, F_ω, F_x

if $\hat{x}^*$ lies in $[a_k,b_k]$ then we accept it as μ_{k+1} otherwise use Golden section.

2.5 Bounding Phase

The reader will notice that all the methods considered so far were interval reduction techniques and that all these methods required an initial interval. The question is now: how do we determine this initial interval? In general we perform a coarse search to *bound* or *bracket* the minimum x^*. This is also called *interval location*. We will discuss two types of interval location:

1) function comparison; and

2) polynomial extrapolation.

Both of these methods are heuristic in nature but are about the best we can provide in terms of an algorithm. We start with a description of Swann's bracketing method.

2.5.1 Function comparison: Swann's Bracketing Method

Given an initial step length Δ and starting point a_1 we check the point a distance Δ away on the right side of a_1. If $f(a_1 + \Delta) < f(a_1)$ then we will move to the right. If on the other hand $f(a_1 + \Delta) > f(a_1)$ then we will move to the left (lets say we move to the right). We now magnify the step size by a magnification factor, say 2, and evaluate the function at the point 2Δ away from the latest point. As soon as any new function evaluation shows an increase we can say that the last two intervals provide a bound or interval of uncertainty for our minimum. An example of the process is shown in Figure 2.13.

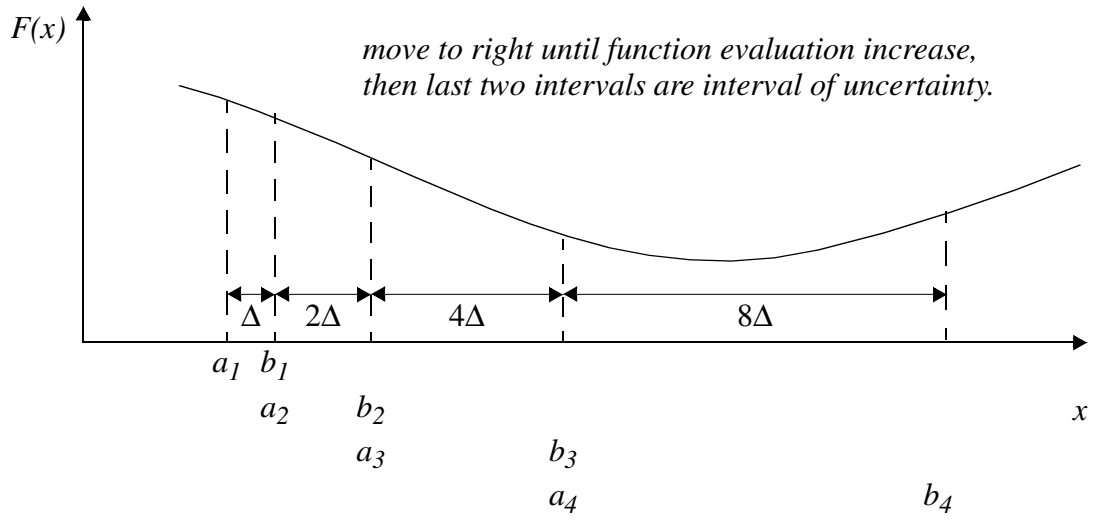

Figure 2.13 Example of Swann's bracketing method, magnification factor is 2.0.

The process can be written in algorithmic form as follows:

Algorithm: Swann's Bracketing Method - based on a heuristic expanding pattern.

1. input: x_0, Δ // initial point and step size
2. set: $x_l = x_0 - |\Delta|$ // lower test point.
3. $x_u = x_0 + |\Delta|$ // upper test point.
4. $f_l = f(x_l)$ // lower function value
5. $f_u = f(x_u)$ // upper function value
6. $f_0 = f(x_0)$ // central function value
7. $i = 1$ // expanding exponent
8. if $f_l \geq f_0 \geq f_u$ // **case 1**: move to the right
9. *loop-up*: $i = i + 1$
10. set: $x_l = x_0$, $x_0 = x_u$, $f_l = f_0$, $f_0 = f_u$
11. $x_u = x_u + 2^i|\Delta|$ // shift up in x by a magnified amount
12. $f_u = f(x_u)$
13. if $f_u < f_0$ go to *loop-up*
14. else output (x_l, x_u)
15. if $f_l \leq f_0 \leq f_u$ // **case 2**: move to the left
16. *loop-down*: $i = i + 1$
17. set: $x_u = x_0$, $x_0 = x_l$, $f_u = f_0$, $f_0 = f_l$
18. $x_l = x_l - 2^i|\Delta|$ // shift down in x by a magnified amount
19. $f_l = f(x_l)$
20. if $f_l < f_0$ go to *loop-down*
21. else output (x_l, x_u)
22. if $f_l \geq f_0 \leq f_u$ // **case 3**: initial interval is bracket
23. output (x_l, x_u)
24. if $f_l \leq f_0 \geq f_u$ *error*: non-unimodal function

2.5.2 Polynomial Extrapolation: Powell's Method

In this method we use the same idea as in polynomial interpolation for interval reduction but we know must extrapolate from the initial starting point. Given a starting point a, and two more points Δ apart we *extrapolate* a polynomial through them (we will use a quadratic function).

Now in this case the fitted quadratic may have either a maximum or minimum and this may be determined by evaluating the second derivative. As before if the polynomial is represented as

$$F(x) = px^2 + qx + r \tag{2.10}$$

then the second derivative is given as

$$F''(x) = G(x) = 2p \tag{2.11}$$

and in terms of the three function values F_a, F_b, F_c, evaluated at $x = a$, b, and c we can evaluate p as

$$p = \frac{(c-b)F_a + (a-c)F_b + (b-a)F_c}{(b-c)(c-a)(a-b)} \quad . \tag{2.12}$$

Now how do we use this to find an interval? The procedure can be described as follows. Starting from three points, a_1, c_1, and b_1, a Δ apart we fit a quadratic to these points, say $F(x)$. If we find that this quadratic has a maximum, that is $p < 0$, then the next point we take is Δmax away from the smallest function value. This is shown in Figure 2.14 where for sake of an example we are assuming that we move to the right. Of course the algorithm will determine the direction of movement.

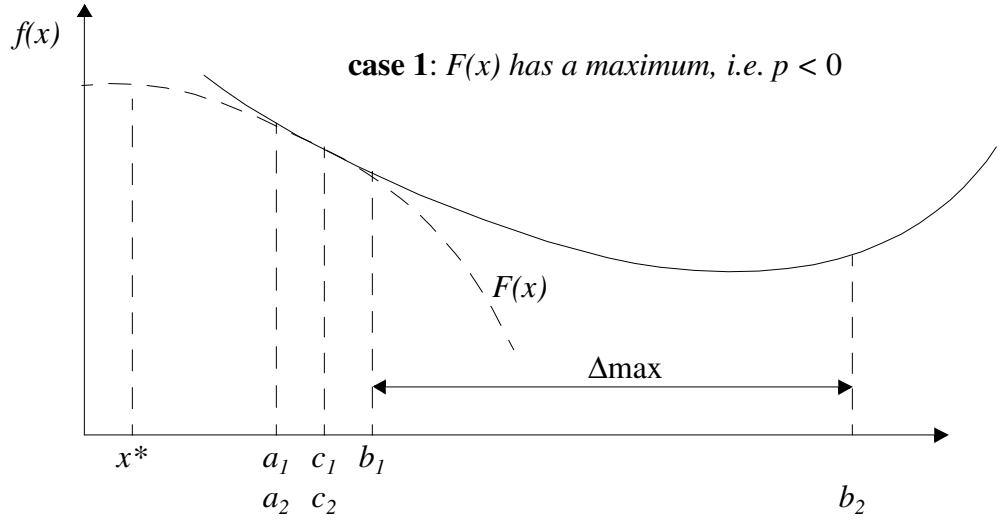

Figure 2.14 Extrapolated polynomial has a maximum therefore use the Δmax.

As can be seen in Figure 2.14, for case 1, we discard the point with smallest function value and take new point Δmax away from b_1. Discarding the lowest function value gives us more of a

chance of fitting a polynomial with a minimum in the next iteration of the algorithm. If the polynomial does have a minimum then we either use the minimum of the polynomial as shown in case 3 of Figure 2.16 or we again use Δmax if the minimum is farther than Δmax from b_1 as shown in case 2 of Figure 2.15. An implementation of this algorithm is shown below.

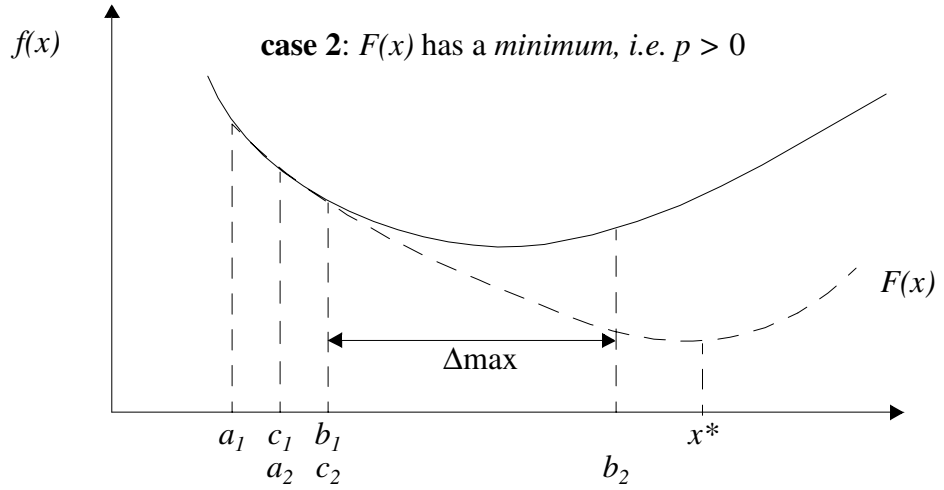

Figure 2.15 Extrapolated polynomial has a minimum but it is too far away so use Δmax.

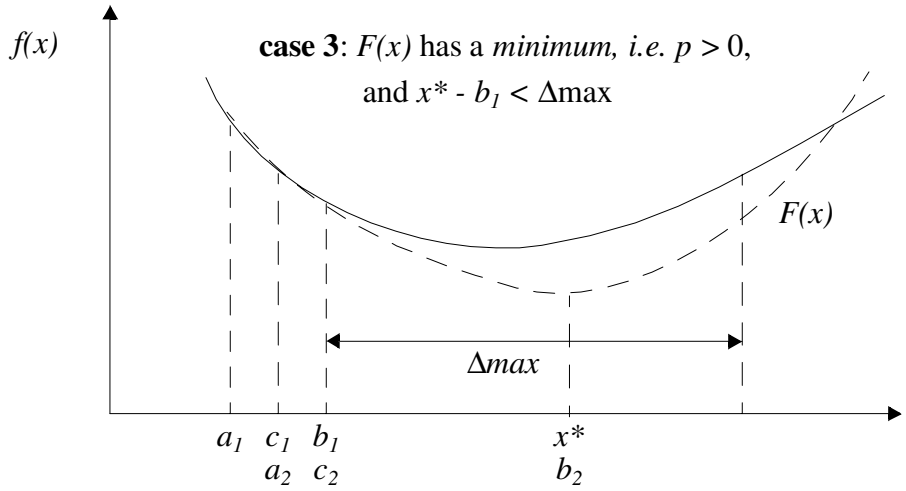

Figure 2.16 Extrapolated polynomial has a minimum and it is closer than Δmax from b_1.

Algorithm: Interval location by Powell's Method

1. input a_1, Δ, Δmax
2. set $c_1 = a_1 + \Delta$, $F_a = F(a_1)$, $F_c = F(c_1)$
3. if $Fa > Fc$ then
4. set $b_1 = a_1 + 2\Delta$, $F_b = F(b_1)$
5. *forward* = true
6. else
7. set $b_1 = c_1$, $c_1 = a_1$, $a_1 = a_1 - \Delta$
8. $F_b = F_c$, $F_c = F_a$, $F_a = F(a_1)$
9. *forward* = false
10. end
11. for $K = 1, 2, \ldots$ repeat
12. set $p = \dfrac{(c_K - b_K)F_a + (a_K - c_K)F_b + (b_K - a_K)F_c}{(b_k - c_K)(c_K - a_K)(a_K - b_K)}$
13. if $p > 0$ then
14. set $x^* = \dfrac{1}{2}\dfrac{(b_K^2 - c_K^2)F_a + (c_k^2 - a_K^2)F_b + (a_K^2 - b_K^2)F_c}{(b_K - c_K)F_a + (c_K - a_K)F_b + (a_K - b_K)F_c}$
15. end if
16. if *forward* then // moving forward
17. if $p \le 0$ then // quadratic has a maximum
18. set $a_{K+1} = a_K$, $b_{K+1} = b_K + \Delta$max
19. $c_{K+1} = c_K$, $F_b = F(b_{K+1})$
20. else
21. if $x^* - b_K > \Delta$max then // quadratic minimum is too far
22. set $b_{K+1} = b_K + \Delta$ max
23. else // quadratic minimum is O.K.
24. set $b_{K+1} = x^*$
25. end
26. set $a_{K+1} = c_K$, $c_{K+1} = b_K$
27. $F_a = F_c$, $F_c = F_b$, $F_b = F(b_{K+1})$
28. end

29. else // moving backward (*forward* = false)
30. if $p \le 0$ then // quadratic has a maximum
31. set $a_{K+1} = a_K - \Delta\max$, $b_{K+1} = b_K$
32. $c_{K+1} = c_K$, $F_a = F(a_{K+1})$
33. else
34. if $a_K - x^* > \Delta\max$ then // quadratic minimum is too far
35. set $a_{K+1} = a_K - \Delta\max$
36. else // quadratic minimum is O.K.
37. set $a_{K+1} = x^*$
38. end
39. set $b_{K+1} = c_K$, $c_{K+1} = a_K$
40. $F_b = F_c$, $F_c = F_a$, $F_a = F(a_{K+1})$
41. end // if $p \le 0$
42. end
43. end until $F_c < F_a$ and $F_c < F_b$