# ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES

*RICHARD P. BRENT*
Thomas J. Watson Research Center
Yorktown Heights, New York

*Prentice-Hall, Inc., Englewood Cliffs, New Jersey*

— *tol*);

# 5

# AN ALGORITHM WITH GUARANTEED CONVERGENCE FOR FINDING A MINIMUM OF A FUNCTION OF ONE VARIABLE

## *Section 1*
## *INTRODUCTION*

A common computational problem is finding an approximation to the minimum or maximum of a real-valued function $f$ in some interval $[a, b]$. This problem may arise directly or indirectly. For example, many methods for minimizing functions $g(\mathbf{x})$ of several variables need to minimize functions of one variable of the form

$$\gamma(\lambda) = g(\mathbf{x}_0 + \lambda \mathbf{s}), \tag{1.1}$$

where $\mathbf{x}_0$ and $\mathbf{s}$ are fixed (a "one-dimensional search" from $\mathbf{x}_0$ in the direction $\mathbf{s}$). In this chapter we give an algorithm which finds an approximate local minimum of $f$ by evaluating $f$ at a small number of points. There is a clear analogy between this algorithm and the algorithm for zero-finding described in Chapter 4 (see Section 4). Unless $f$ is unimodal (Section 3), the local minimum may not be the global minimum of $f$ in $[a, b]$, and the problem of finding global minima is left until Chapter 6.

The algorithm described in this chapter could be used to solve the problem (1.1), but it would be more economical to use special algorithms which make use of any extra information which is available (e.g., estimates of the second derivative of $\gamma$), and which do not attempt to find the minimum very accurately. This is discussed in Chapter 7. Thus, a more likely practical use for our algorithm is to find accurate minima of naturally arising functions of one variable.

In Section 2 we consider the effect of rounding errors on any minimization algorithm based entirely on function evaluations. Unimodality is defined in Section 3, and we also define "$\delta$-unimodality" in an attempt to explain why methods like golden section search work even for functions which are not quite unimodal (because of rounding errors in their computation, for example). In Sections 4 and 5 we describe a minimization algorithm analogous to the zero-finding algorithm of Chapter 4, and some numerical results are given in Section 6. Finally, some possible extensions are described in Section 7, and an ALGOL 60 procedure is given in Section 8.

### *Reduction to a zero-finding problem*

If $f$ is differentiable in $[a, b]$, a necessary condition for $f$ to have a local minimum at an interior point $\mu \in (a, b)$ is

$$f'(\mu) = 0. \tag{1.2}$$

There is also the possibility that the minimum is at $a$ or $b$: for example, this is true if $f'$ does not change sign on $[a, b]$. If we are prepared to check for this possibility, one approach is to look for zeros of $f'$. If $f'$ has different signs at $a$ and $b$, then the algorithm of Chapter 4 may be used to approximate a point $\mu$ satisfying (1.2).

Since $f'$ vanishes at any stationary point of $f$, it is possible that the point found is a maximum, or even an inflexion point, rather than a minimum. Thus, it is necessary to check whether the point found is a true minimum, and continue the search in some way if it is not.

If it is difficult or impossible to compute $f'$ directly, we could approximate $f'$ numerically (e.g., by finite differences), and search for a zero of $f'$ as above. However, a method which does not need $f'$ seems more natural, and could be preferred for the following reasons:

1. It may be difficult to approximate $f'$ accurately because of rounding errors;
2. A method which does not need $f'$ may be more efficient (see below); and
3. Whether $f'$ can be computed directly or not, a method which avoids difficulty with maxima and inflexion points is clearly desirable.

### *Jarratt's method*

Jarratt (1967) suggests a method, using successive parabolic interpolation, which is a special case of the iteration analyzed in Chapter 3. With arbitrary starting points Jarratt's method may diverge, or converge to a maximum or inflexion point, but this defect need not be fatal if the method is used in combination with a safe method such as golden section search, in the

same way that we used a combination of successive linear interpolation and bisection for finding a zero. Theorem 3.5.1 shows that, if $f$ has a Lipschitz continuous second derivative which is positive at an interior minimum $\mu$, then Jarratt's method gives superlinear convergence to $\mu$ with weak order at least $\beta_2 = 1.3247\ldots$ (see Definitions 3.2.1 and 3.5.1), provided the initial approximation is good and rounding errors are negligible.

Let us compare Jarratt's method with one of the alternatives: estimating $f'$ by finite differences, and then using successive linear interpolation to find a zero of $f'$. (This process may also diverge, or converge to a maximum.) Suppose that $f''(\mu) > 0$ and $f^{(3)}(\mu) \neq 0$, to avoid exceptional cases (see Sections 3.6, 3.7, and 4.2). Since at least two function evaluations are needed to estimate $f'$ at any point, and $\sqrt{1.618\ldots} = 1.272\ldots < 1.324\ldots$, Jarratt's method has a slightly higher order of convergence. The comparison is similar to that between Newton's method and successive linear interpolation: see Section 4.3 and Ostrowski (1966).

## Section 2
## FUNDAMENTAL LIMITATIONS BECAUSE OF ROUNDING ERRORS

Suppose that $f \in LC^2[a, b; M]$ has a minimum at $\mu \in (a, b)$. Since $f'(\mu) = 0$, Lemma 2.3.1 gives, for $x \in [a, b]$,

$$f(x) = f_0 + \frac{1}{2} f_0''(x - \mu)^2 + \frac{m_x}{6}(x - \mu)^3, \tag{2.1}$$

where $|m_x| \leq M$, $f_0 = f(\mu)$, and $f_0'' = f''(\mu)$. Because of rounding errors, the best that can be expected if single-precision floating-point numbers are used is that the computed value $fl(f(x))$ of $f(x)$ satisfies the (nearly attainable) bound

$$fl(f(x)) = f(x)(1 + \epsilon_x), \tag{2.2}$$

where

$$|\epsilon_x| \leq \epsilon, \tag{2.3}$$

and $\epsilon$ is the relative machine precision (see Section 4.2). The error bound is unlikely to be as good as this unless $f$ is a very simple function, or is evaluated using double-precision and then rounded or truncated to single-precision.

Let $\delta$ be the largest number such that, according to equations (2.2) and (2.3), it is possible that

$$fl(f(\mu + \delta)) \leq f_0. \tag{2.4}$$

It is unreasonable to expect any minimization procedure, based on single-precision evaluations of $f$, to return an approximation $\hat{\mu}$ to $\mu$ with a guar-

anteed upper bound for $|\hat{\mu} - \mu|$ less than $\delta$. This is so regardless of whether the computed values of $f$ are used directly, as in Jarratt's method, or indirectly, as in the other method suggested in Section 1. The reason is simply that the minimum of the computed function $fl(f(x))$ may lie up to a distance $\delta$ from the minimum $\mu$ of $f(x)$: see Diagram 2.1.

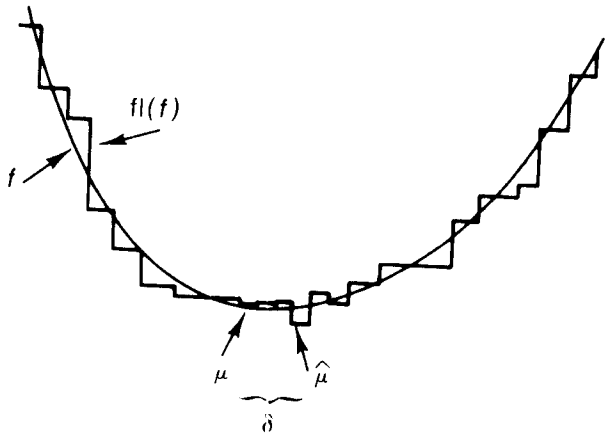


***DIAGRAM 2.1*** **The effect of rounding errors**

If $f_0'' > 0$, equations (2.1) to (2.4) give

$$\delta \geq \sqrt{\frac{2|f_0|\epsilon}{f_0''}}\left(1 - \epsilon - \frac{M\delta}{6f_0''}\right). \tag{2.5}$$

Thus, if $\mu \neq 0$ and the term $M\delta/(6f_0'')$ is negligible, an upper bound for the relative error $|(\hat{\mu} - \mu)/\mu|$ could hardly be less than $[2|f_0|\epsilon/(\mu^2 f_0'')]^{1/2}$, and full single-precision accuracy in $\hat{\mu}$ is unlikely unless $|f_0|/(\mu^2 f_0'')$ is of order $\epsilon$ or less, although $fl(f(\hat{\mu}))$ may agree with $f(\mu)$ to full single-precision accuracy. (See also Pike, Hill, and James (1967).)

If $f'$ has a simple analytic representation, then it may be easy to compute $f'$ accurately. For example, perhaps

$$fl(f'(x)) = f'(x(1 + \epsilon_x'))(1 + \epsilon_x''), \tag{2.6}$$

where $|\epsilon_x'| \leq \epsilon$ and $|\epsilon_x''| \leq \epsilon$, so we can expect to find a zero of $f'$ with a relative error bounded by $\epsilon$ (see Lancaster (1966) and Ostrowski (1967b)). If (2.6) holds it might be worthwhile to use the algorithm described in Chapter 4 to search for a zero of $f'$, or at least use it to refine the approximation $\hat{\mu}$ given by a procedure using only evaluations of $f$. However, this is not so if $f'$ has to be approximated by differences, for then (2.6) cannot be expected to hold.

Even if $f(x)$ is a unimodal function, the computed approximation $fl(f(x))$ will not be unimodal: $fl(f(x))$ must be constant over small intervals of real numbers $x$ which have the same floating-point approximation $fl(x)$. In the next section we define "$\delta$-unimodality" to circumvent this difficulty.

From now on, we consider the problem of approximating the minimum of the *computed* function, or, equivalently, we ignore rounding errors in the computation of $f$. The user should bear in mind that the minimum of the computed function may differ from the minimum that he is really interested in by as much as $\delta$ (see equation (2.5) above). There is no point in wasting function evaluations by finding the minimum of the computed function to excessive accuracy, and our procedure *localmin* (Section 8) should not be called with the parameter *eps* much less than $[2|f_0|\epsilon/(\mu^2 f_0'')]^{1/2}$.

## Section 3
## UNIMODALITY AND δ-UNIMODALITY

There are several different definitions of a unimodal function in the literature. One source of confusion is that the definition depends on whether the function is supposed to have a unique minimum or a unique maximum (we consider minima). Kowalik and Osborne (1968) say that $f$ is unimodal on $[a, b]$ if $f$ has only one stationary value on $[a, b]$. This definition has two disadvantages. First, it is meaningless unless $f$ is differentiable on $[a, b]$, but we would like to say that $|x|$ is unimodal on $[-1, 1]$. Second, functions which have inflexion points with a horizontal tangent are prohibited, but we would like to say that $f(x) = x^6 - 3x^4 + 3x^2$ is unimodal on $[-2, 2]$ (here $f'(\pm 1) = f''(\pm 1) = 0$).

Wilde (1964) gives another definition: $f$ is unimodal on $[a, b]$ if, for all $x_1, x_2 \in [a, b]$,

$$x_1 < x_2 \supset (x_2 < x^* \supset f(x_1) > f(x_2)) \wedge (x_1 > x^* \supset f(x_1) < f(x_2)), \tag{3.1}$$

where $x^*$ is a point at which $f$ attains its least value in $[a, b]$. (We have reversed some of Wilde's inequalities as he considers maxima rather than minima.) Wilde's definition does not assume differentiability, or even continuity, but to verify that a function $f$ satisfies (3.1) we need to know the point $x^*$ (and such a point must exist). Hence, we prefer the following definition, which is nearly equivalent to Wilde's (see Lemma 3.1), but avoids any reference to the point $x^*$. The definition is not as complicated as it looks: it merely says that $f$ cannot have a "hump" between any two points $x_0$ and $x_2$ in $[a, b]$. Two possible configurations of the points $x_0$, $x_1$, $x_2$, and $x^*$ in (3.1) and (3.2) are illustrated in Diagram 3.1.

**DEFINITION 3.1**

$f$ is *unimodal* on $[a, b]$ if, for all $x_0$, $x_1$ and $x_2 \in [a, b]$,

$$\begin{aligned} x_0 < x_1 \wedge x_1 < x_2 \supset (f(x_0) \leq f(x_1) \supset f(x_1) < f(x_2)) \wedge \\ (f(x_1) \geq f(x_2) \supset f(x_0) > f(x_1)). \end{aligned} \tag{3.2}$$

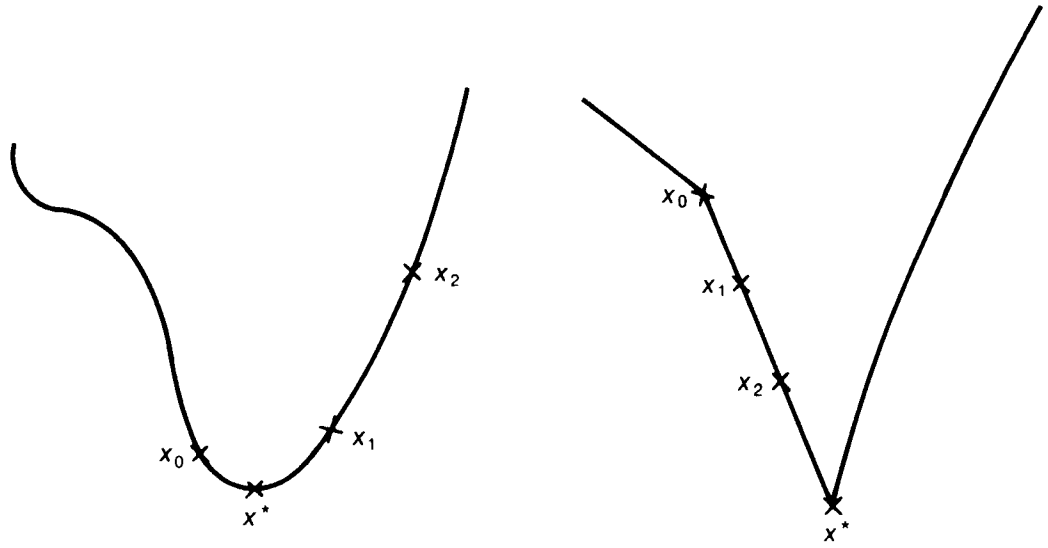


*DIAGRAM 3.1* **Unimodal functions**

**LEMMA 3.1**

If a point $x^*$ at which $f$ attains its minimum in $[a, b]$ exists, then Wilde's definition of unimodality and Definition 3.1 are equivalent.

Proof

Suppose that $f$ is unimodal according to Definition 3.1. If $x_1 < x_2$ and $x_2 < x^*$, take $x_0' = x_1$, $x_1' = x_2$, and $x_2' = x^*$. Since $f$ attains its least value at $x^*$,

$$f(x_1') \geq f(x^*) = f(x_2'), \tag{3.3}$$

so equation (3.2) with primed variables gives

$$f(x_0') > f(x_1'), \tag{3.4}$$

and thus

$$f(x_1) > f(x_2). \tag{3.5}$$

Similarly, if $x_1 < x_2$ and $x_1 > x^*$, equation (3.2) gives

$$f(x_1) < f(x_2). \tag{3.6}$$

Thus, from (3.5) and (3.6), equation (3.1) holds.

Conversely, suppose that (3.1) holds and $x_0 < x_1 < x_2$. If $f(x_0) \leq f(x_1)$ then there are three possibilities, depending on the position of $x^*$:

1. $x_1 > x^*$. Thus, by (3.1),
$$f(x_1) < f(x_2). \tag{3.7}$$

2. $x_1 = x^*$. Take $x_1' = \frac{1}{2}(x_1 + x_2)$ and $x_2' = x_2$.
Since $x^* < x_1' < x_2'$, equation (3.1) with primed variables gives
$$f(x_1') < f(x_2'), \tag{3.8}$$
so
$$f(x_1) = f(x^*) \leq f(x_1') < f(x_2') = f(x_2). \tag{3.9}$$

3. $x_1 < x^*$. Take $x_1' = x_0$ and $x_2' = x_1$. Since $x_1' < x_2' < x^*$, equation (3.1) gives $f(x_1') > f(x_2')$, contradicting the assumption that $f(x_0) \leq f(x_1)$. Hence case 3 is impossible and, by (3.7) and (3.9), we always have $f(x_1) < f(x_2)$.

Similarly, if $f(x_1) \geq f(x_2)$ then $f(x_0) > f(x_1)$, so equation (3.2) holds, and the proof is complete.

A simple corollary of Lemma 3.1 is that, if $f$ is continuous, then Wilde's definition of unimodality and ours are equivalent. For arbitrary $f$ the definitions are not equivalent. For example,

$$f(x) = \begin{cases} 1 - x & \text{if} \quad x \leq 0, \\ x & \text{if} \quad x > 0 \end{cases} \tag{3.10}$$

is unimodal on $[-1, 1]$ by our definition, but not by Wilde's, for $x^*$ does not exist.

The following theorem gives a simple characterization of unimodality. There is no assumption that $f$ is continuous. Since a strictly monotonic function (e.g., $x^3$) may have stationary points, the theorem shows that both our definition and Wilde's are essentially different from Kowalik and Osborne's, even if $f$ is continuously differentiable. (Although this point is obvious, it is sometimes overlooked! See also Corollary 3.3.)

**THEOREM 3.1**

$f$ is unimodal on $[a, b]$ (according to Definition 3.1) iff, for some (unique) $\mu \in [a, b]$, either $f$ is strictly monotonic decreasing on $[a, \mu)$ and strictly monotonic increasing on $[\mu, b]$, or $f$ is strictly monotonic decreasing on $[a, \mu]$ and strictly monotonic increasing on $(\mu, b]$.

The theorem is a special case of Theorem 3.2 below, so the proof is omitted. The following corollaries are immediate.

**COROLLARY 3.1**

If $f$ is unimodal on $[a, b]$, then $f$ attains its least value at most once on $[a, b]$. (If $f$ attains its least value, then it must attain it at the point $\mu$ given by Theorem 3.1.)

**COROLLARY 3.2**

If $f$ is unimodal and continuous on $[a, b]$, then $f$ attains its least value exactly once on $[a, b]$.

**COROLLARY 3.3**

If $f \in C^1[a, b]$ then $f$ is unimodal iff, for some $\mu \in [a, b]$, $f' < 0$ almost everywhere on $[a, \mu]$ and $f' > 0$ almost everywhere on $[\mu, b]$. (Note that $f'$ may vanish at a finite number of points.)

### *Fibonacci and golden section search*

If $f$ is unimodal on $[a, b]$, then the minimum of $f$ (or, if the minimum is not attained, the point $\mu$ given by Theorem 3.1) can be located to any desired accuracy by the well-known methods of Fibonacci search or golden section search. The reader is referred to Wilde (1964) for an excellent description of these methods. (See also Boothroyd (1965a, b), Johnson (1955), Krolak (1968), Newman (1965), Pike and Pixner (1967), and Witzgall (1969).) Care should be taken to ensure that the coordinates of the points at which $f$ is evaluated are computed in a numerically stable way (see Overholt (1965)). Fibonacci and golden section search, as well as similar but less efficient methods, are based on the following result, which shows how an interval known to contain $\mu$ may be reduced in size.

**COROLLARY 3.4**

Suppose that $f$ is unimodal on $[a, b]$, $\mu$ is the point given by Theorem 3.1, and $a \leq x_1 < x_2 \leq b$. If $f(x_1) \leq f(x_2)$ then $\mu \leq x_2$, and if $f(x_1) \geq f(x_2)$ then $\mu \geq x_1$.

Proof

If $x_2 < \mu$ then, by Theorem 3.1, $f(x_1) > f(x_2)$. Thus, if $f(x_1) \leq f(x_2)$ then $\mu \leq x_2$. The other half follows similarly.

If the reader is prepared to ignore the problem of computing unimodal functions using limited-precision arithmetic, he may skip the rest of this section.

### *δ-unimodality*

We pointed out at the end of Section 2 that functions computed using limited-precision arithmetic are not unimodal. Thus, the theoretical basis for Fibonacci search and similar methods is irrelevant, and it is not clear that these methods will give even approximately correct results in the presence of rounding errors. To analyze this problem, we generalize the idea of unimodality to $\delta$-unimodality. Intuitively, $\delta$ is a nonnegative number such that Fibonacci or golden section search will give correct results, even though $f$ is not necessarily unimodal (unless $\delta = 0$), provided that the distance between points at which $f$ is evaluated is always greater than $\delta$. The results of Section 2 indicate how large $\delta$ is likely to be in practice. (Our aim differs from that of Richman (1968) in defining the $\epsilon$-calculus, for he is interested in properties that hold as $\epsilon \to 0$.) For another approach to the problem of rounding errors, see Overholt (1967).

In the remainder of this section, $\delta$ is a fixed nonnegative number. As well as $\delta$-unimodality, we need to define $\delta$-monotonicity. If $\delta = 0$ then

$\delta$-unimodality and $\delta$-monotonicity reduce to unimodality (Definition 3.1) and monotonicity.

**DEFINITION 3.2**

Let $I$ be an interval and $f$ a real-valued function on $I$. We say that $f$ is *strictly $\delta$-monotonic increasing* on $I$ if, for all $x_1, x_2 \in I$,

$$x_1 + \delta < x_2 \supset f(x_1) < f(x_2). \tag{3.11}$$

As an abbreviation, we shall write simply "$f$ is $\delta$-$\uparrow$ on $I$". Strictly $\delta$-monotonic decreasing functions (abbreviated $\delta$-$\downarrow$) are defined in the obvious way.

**DEFINITION 3.3**

Let $I$ be an interval and $f$ a real-valued function on $I$. We say that $f$ is *$\delta$-unimodal* on $I$ if, for all $x_0, x_1, x_2 \in I$,

$$x_0 + \delta < x_1 \wedge x_1 + \delta < x_2 \supset (f(x_0) \leq f(x_1) \supset f(x_1) < f(x_2)) \wedge (f(x_1) \geq f(x_2) \supset f(x_0) > f(x_1)). \tag{3.12}$$

The following theorem gives a characterization of $\delta$-unimodal functions. It reduces to Theorem 3.1 if $\delta = 0$.

**THEOREM 3.2**

$f$ is $\delta$-unimodal on $[a, b]$ iff there exists $\mu \in [a, b]$ such that either $f$ is $\delta$-$\downarrow$ on $[a, \mu)$ and $\delta$-$\uparrow$ on $[\mu, b]$, or $f$ is $\delta$-$\downarrow$ on $[a, \mu]$ and $\delta$-$\uparrow$ on $(\mu, b]$. Furthermore, if $f$ is $\delta$-unimodal on $[a, b]$, then there is a unique interval $[\mu_1, \mu_2] \subseteq [a, b]$ such that the points $\mu$ with the above properties are precisely the elements of $[\mu_1, \mu_2]$, and $\mu_2 \leq \mu_1 + \delta$.

Proof

Suppose $\mu$ exists so that $f$ is $\delta$-$\downarrow$ on $[a, \mu)$ and $\delta$-$\uparrow$ on $[\mu, b]$. Take any $x_0, x_1, x_2$ in $[a, b]$ with $x_0 + \delta < x_1$ and $x_1 + \delta < x_2$. If $f(x_0) \leq f(x_1)$ then, since $f$ is $\delta$-$\downarrow$ on $[a, \mu)$, $\mu \leq x_1$. As $f$ is $\delta$-$\uparrow$ on $[\mu, b)$, it follows that $f(x_1) < f(x_2)$. The other cases are similar, so $f$ is $\delta$-unimodal.

Conversely, suppose that $f$ is $\delta$-unimodal on $[a, b]$. Let

$$\mu_1 = \inf\{x \in [a, b] \mid f \text{ is } \delta\text{-}\uparrow \text{ on } [x, b]\}, \tag{3.13}$$

(so $\mu_1 \leq \max(a, b - \delta)$), and

$$\mu_2 = \sup\{x \in [a, b] \mid f \text{ is } \delta\text{-}\downarrow \text{ on } [a, x]\}, \tag{3.14}$$

(so $\mu_2 \geq \min(a + \delta, b)$).

It is immediate from the definitions (3.13) and (3.14) that $f$ is $\delta$-$\uparrow$ on $(\mu_1, b]$ and $f$ is $\delta$-$\downarrow$ on $[a, \mu_2)$. We shall show that

$$\mu_1 \leq \mu_2. \tag{3.15}$$

Suppose, by way of contradiction, that

$$\mu_1 > \mu_2. \tag{3.16}$$

This implies that $\mu_1 > a$ and $\mu_2 < b$. From the definitions of $\mu_1$ and $\mu_2$, there are points $x'$ and $x''$, with

$$\mu_2 \leq x'' < \left(\frac{\mu_1 + \mu_2}{2}\right) < x' \leq \mu_1, \tag{3.17}$$

such that $f$ is not $\delta$-$\uparrow$ on $[x', b]$ and $f$ is not $\delta$-$\downarrow$ on $[a, x'']$. Thus, there are points $y'$, $y''$, $z'$, $z''$ in $[a, b]$ such that

$$z'' + \delta < y'' \leq x'' < x' \leq y' < z' - \delta, \tag{3.18}$$

$$f(z'') \leq f(y''), \tag{3.19}$$

and

$$f(y') \geq f(z'). \tag{3.20}$$

Let $x_0 = z''$, $x_2 = z'$, and

$$x_1 = \begin{cases} y' & \text{if } f(y') \geq f(y''), \\ y'' & \text{otherwise.} \end{cases} \tag{3.21}$$

From relations (3.18) to (3.21), the points $x_0$, $x_1$, and $x_2$ contradict $\delta$-unimodality (equation (3.12)). Thus (3.16) is impossible, (3.15) must hold, and $[\mu_1, \mu_2]$ is nonempty.

Choose any $\mu$ in $[\mu_1, \mu_2]$. From the definitions of $\mu_1$ and $\mu_2$, $f$ is $\delta$-$\downarrow$ on $[a, \mu)$ and $\delta$-$\uparrow$ on $(\mu, b]$. Suppose, if it is possible, that $f$ is neither $\delta$-$\downarrow$ on $[a, \mu]$ nor $\delta$-$\uparrow$ on $[\mu, b]$. Then there are points $y_1$ and $y_2$, in $[a, b]$, such that

$$y_2 + \delta < \mu < y_1 - \delta, \tag{3.22}$$

$$f(y_1) \leq f(\mu), \tag{3.23}$$

and

$$f(y_2) \leq f(\mu). \tag{3.24}$$

Thus, the points $y_2$, $\mu$, and $y_1$ contradict the $\delta$-unimodality of $f$, so $f$ is either $\delta$-$\downarrow$ on $[a, \mu]$ or $\delta$-$\uparrow$ on $[\mu, b]$. This completes the proof of the first part of the theorem.

Finally, by the definitions (3.13) and (3.14), the set of points $\mu$ satisfying the conditions of the theorem is precisely $[\mu_1, \mu_2]$. Since $f$ is both $\delta$-$\uparrow$ and $\delta$-$\downarrow$ on $(\mu_1, \mu_2)$, we have $\mu_2 \leq \mu_1 + \delta$, and the proof is complete.

### Remarks

The interval $[\mu_1, \mu_2]$ depends on $\delta$. Suppose that $f$ attains its minimum in $[a, b]$ at $\bar{\mu}$. By Theorem 3.2, $f$ is $\delta$-$\uparrow$ on $(\mu_1, b]$ and $\delta$-$\downarrow$ on $[a, \mu_2)$, so $\bar{\mu} \in [\mu_2 - \delta, \mu_1 + \delta]$, an interval of length at most $2\delta$.

As an example, consider

$$f(x) = x^2 + g(x) \tag{3.25}$$

on $[-1, 1]$, where $g$ is any function (not necessarily continuous) with $|g(x)| \leq \epsilon$, and $\epsilon \geq 0$. Since $f(x)$ is bounded above and below by the unimodal functions $x^2 + \epsilon$ and $x^2 - \epsilon$, we see that $f$ is $\delta$-unimodal for any $\delta \geq \sqrt{2\epsilon}$. In a practical case $\epsilon$ might be a small multiple of the relative machine precision, and the fact that the least $\delta$ for which $f$ is $\epsilon$-unimodal is of order $\epsilon^{1/2}$, rather than $\epsilon$, is to be expected from the discussion in Section 2.

The following theorem is a generalization of Corollary 3.4 (which is just the special case $\delta = 0$), and shows why methods like Fibonacci search and golden section search work on $\delta$-unimodal functions while the distance between points at which $f$ is evaluated is greater than $\delta$.

**THEOREM 3.3**

Suppose that $f$ is $\delta$-unimodal on $[a, b]$, $\mu_1$ and $\mu_2$ are the points given by Theorem 3.2, $x_1$ and $x_2$ are in $[a, b]$, and $x_1 + \delta < x_2$. If $f(x_1) \leq f(x_2)$ then $\mu_2 \leq x_2$, and if $f(x_1) \geq f(x_2)$ then $\mu_1 \geq x_1$.

Proof

If $x_2 < \mu_2$ then $f(x_1) > f(x_2)$ for, by Theorem 3.2 with $\mu = \mu_2$, $f$ is $\delta$-$\downarrow$ on $[a, \mu_2)$. Hence, if $f(x_1) \leq f(x_2)$ then $\mu_2 \leq x_2$. The second half is similar.

Remarks

Theorems 3.2 and 3.3 show that, provided $\delta$ is known, methods like Fibonacci search and golden section search can locate the interval $[\mu_1, \mu_2]$ in an interval of length as close to $\delta$ as desired. Since the minimum $\bar{\mu} \in [\mu_2 - \delta, \mu_1 + \delta]$ (see the remarks above), this means that $\bar{\mu}$ can be located in an interval of length as close to $3\delta$ as desired.

In practice $f$ may be $\delta$-unimodal for all $\delta \geq \delta_0$, but a sharp upper bound for $\delta_0$ may be difficult to obtain. If the usual golden section search method is used, giving a nested sequence of intervals $I_j$ with limit $\hat{\mu}$, then Theorem 3.3 shows that $[\mu_1, \mu_2] \subseteq I_j$ as long as the two function evaluations giving $I_j$ were at points separated by more than $\delta_0$. The smallest such interval $I_j$ has length no greater than $(2 + \sqrt{5})\delta_0$, so

$$|\hat{\mu} - \bar{\mu}| \leq (3 + \sqrt{5})\delta_0 \simeq 5.236\delta_0. \tag{3.26}$$

Thus, golden section search gives an approximation $\hat{\mu}$ which is nearly as good as could be expected if we knew $\delta_0$. This may be regarded as a justification for using golden section or Fibonacci search to approximate minima of functions which, because of rounding errors, are only "approximately" unimodal.

## Section 4
## AN ALGORITHM ANALOGOUS TO DEKKER'S ALGORITHM

For finding a zero of a function $f$, the bisection process has the advantage that linear convergence is guaranteed, because the interval known to contain a zero is halved at each evaluation of $f$ after the first. However, if $f$ is sufficiently smooth and we have a good initial approximation to a simple zero, then a process with superlinear convergence will be much faster than bisection. This is the motivation for the algorithm, described in Chapter 4, which combines bisection and successive linear interpolation in a way which retains the advantages of both.

There is a clear analogy between methods for finding a minimum and for finding a zero. The Fibonacci and golden section search methods have guaranteed linear convergence, and correspond to bisection. Processes like successive parabolic interpolation, which do not always converge, but under certain conditions converge superlinearly, correspond to successive linear interpolation. In this section we describe an algorithm which combines golden section search and successive parabolic interpolation. The analogy with the algorithm of Chapter 4 is illustrated below.

| | *Zeros* | | *Extrema* |
|---|---|---|---|
| *Linear convergence* | Bisection | ⟷ | Golden section search |
| | ↕ | | ↕ |
| *Superlinear convergence* | Successive linear interpolation | ⟷ | Successive parabolic interpolation |

Many more or less *ad hoc* algorithms have been proposed for one-dimensional minimization, particularly as components of $n$-dimensional minimization algorithms. See Box, Davies, and Swann (1969); Flanagan, Vitale, and Mendelsohn (1969); Fletcher and Reeves (1964); Jacoby, Kowalik, and Pizzo (1971); Kowalik and Osborne (1968); Pierre (1969); Powell (1964); etc. The algorithm presented here might be regarded as an unwarranted addition to this list, but it seems to be more natural than these algorithms, which involve arbitrary prescriptions like "if . . . fails then halve the step-size and try again". Of course, our algorithm is not quite free of arbitrary prescriptions either; a more objective criticism of the *ad hoc* algorithms is that for many of them convergence to a local minimum in a reasonable number of function evaluations cannot be guaranteed, and, for the exceptions, the asymptotic rate of convergence (when $f$ is sufficiently smooth) is less than for our algorithm (Section 5). Note that we do not claim that our algorithm is suitable for use in an $n$-dimensional minimization procedure: an *ad hoc* algorithm may be more efficient (see Sections 7.6 and 7.7).

### A description of the algorithm

Here we give an outline which should make the main ideas of the algorithm clear. For questions of detail the reader should refer to Section 8, where the algorithm is described formally by the ALGOL 60 procedure *localmin*.

The algorithm finds an approximation to the minimum of a function $f$ defined on the interval $[a, b]$. Unless $a$ is very close to $b$, $f$ is never evaluated at the endpoints $a$ and $b$, so $f$ need only be defined on $(a, b)$, and if the minimum is actually at $a$ or $b$ then an interior point distant no more than $2tol$ from $a$ or $b$ will be returned, where $tol$ is a tolerance (see equation (4.2) below). The minimum found may be local, but non-global, unless $f$ is $\delta$-unimodal for some $\delta < tol$.

At a typical step there are six significant points $a$, $b$, $u$, $v$, $w$, and $x$, not all distinct. The positions of these points change during the algorithm, but there should be no confusion if we omit subscripts. Initially $(a, b)$ is the interval on which $f$ is defined, and

$$v = w = x = a + \left(\frac{3 - \sqrt{5}}{2}\right)(b - a). \tag{4.1}$$

The magic number $(3 - \sqrt{5})/2 = 0.381966\ldots$ is rather arbitrarily chosen so that the first step is the same as for a golden section search.

At the start of a cycle (label "loop" of procedure *localmin*) the points $a$, $b$, $u$, $v$, $w$, and $x$ always serve as follows: a local minimum lies in $[a, b]$; of all the points at which $f$ has been evaluated, $x$ is the one with the least value of $f$, or the point of the most recent evaluation if there is a tie; $w$ is the point with the next lowest value of $f$; $v$ is the previous value of $w$; and $u$ is the last point at which $f$ has been evaluated (undefined the first time). One possible configuration is shown in Diagram 4.1.

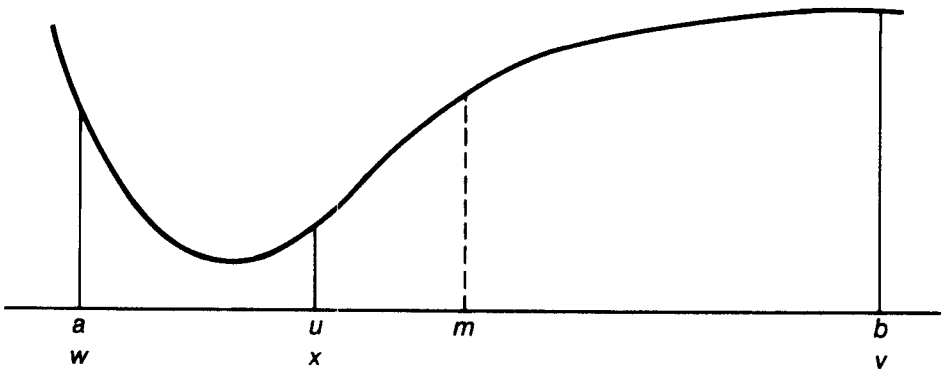


***DIAGRAM 4.1*** **A possible configuration**

As in procedure *zero* (Chapter 4), the tolerance is a combination of a relative and an absolute tolerance. If

$$tol = eps\,|x| + t, \tag{4.2}$$

then the point $x$ returned approximates a minimum to an accuracy of $2tol + \delta < 3tol$, provided $f$ is $\delta$-unimodal near $x$ and $\delta < tol$. The user must provide the positive parameters *eps* and $t$. In view of the discussion in Section 2, it is generally unreasonable to take *eps* much less than $\epsilon^{1/2}$, where $\epsilon$ is the machine-precision (see Section 4.2). The parameter $t$ should be positive in case the minimum is at 0. It is possible that the error may exceed $2tol + \delta$ because of the effect of rounding errors in determining if the stopping criterion is satisfied, but the additional error is negligible if *eps* is of order $\epsilon^{1/2}$ or greater.

Let $m = \frac{1}{2}(a + b)$ be the midpoint of the interval known to contain the minimum. If $|x - m| \leq 2tol - \frac{1}{2}(b - a)$, i.e., if $\max(x - a, b - x) \leq 2tol$, then the procedure terminates with $x$ as the approximate position of the minimum. Otherwise, numbers $p$ and $q (q \geq 0)$ are computed so that $x + p/q$ is the turning point of the parabola passing through $(v, f(v))$, $(w, f(w))$, and $(x, f(x))$. If two or more of these points coincide, or if the parabola degenerates to a straight line, then $q = 0$.

$p$ and $q$ are given by

$$p = \pm[(x - v)^2(f(x) - f(w)) - (x - w)^2(f(x) - f(v))] \tag{4.3}$$

$$= \pm(x - v)(x - w)(w - v)\{(x - w)f[v, w, x] + f[w, x]\}, \tag{4.4}$$

and

$$q = \mp 2[(x - v)(f(x) - f(w)) - (x - w)(f(x) - f(v))] \tag{4.5}$$

$$= \mp 2(x - v)(x - w)(w - v)f[v, w, x]. \tag{4.6}$$

From (4.4) and (4.6), the correction $p/q$ should be small if $x$ is close to a minimum where the second derivative is positive, so the effect of rounding errors in computing $p$ and $q$ is minimized. (Golub and Smith (1967) compute a correction to $\frac{1}{2}(v + w)$ for the same reason.)

As in procedure *zero*, let $e$ be the value of $p/q$ at the second-last cycle. If $|e| \leq tol$, $q = 0$, $x + p/q \notin (a, b)$, or $|p/q| \geq \frac{1}{2}|e|$, then a "golden section" step is performed, i.e., the next value of $u$ is

$$u = \begin{cases} \left(\frac{\sqrt{5} - 1}{2}\right)x + \left(\frac{3 - \sqrt{5}}{2}\right)a & \text{if } x \geq m, \\ \left(\frac{\sqrt{5} - 1}{2}\right)x + \left(\frac{3 - \sqrt{5}}{2}\right)b & \text{if } x < m. \end{cases} \tag{4.7}$$

(If the next $k$ steps are golden section steps, then this is the limit of the optimal choice as $k \to \infty$: see Witzgall (1969).) Otherwise $u$ is taken as $x + p/q$ (a "parabolic interpolation" step), except that the distances $|u - x|$, $u - a$, and $b - u$ must be at least $tol$. Then $f$ is evaluated at the new point $u$, the points $a$, $b$, $v$, $w$, and $x$ are updated as necessary, and the cycle is repeated (the procedure returns to the label "loop"). We see that $f$ is never evaluated at two points closer together than $tol$, so $\delta$-unimodality for some $\delta < tol$ is enough to ensure that the global minimum is found to an accuracy of $2tol + \delta$ (see Theorem 3.3 and the following remarks).

Typically the algorithm terminates in the following way: $x = b - tol$ (or, symmetrically, $a + tol$) after a parabolic interpolation step has been performed with the condition $|u - x| \geq tol$ enforced. The next parabolic interpolation point lies very close to $x$ and $b$, so $u$ is forced to be $x - tol$. If $f(u) > f(x)$ then $a$ moves to $u$, $b - a$ becomes $2tol$, and the termination criterion is satisfied (see Diagram 4.2). Note that two consecutive steps of $tol$ are done just before termination. If a golden section search were done whenever the last, rather than second-last, value of $|p/q|$ was $tol$ or less, then termination with two consecutive steps of $tol$ would be prevented, and unnecessary golden section steps would be performed.

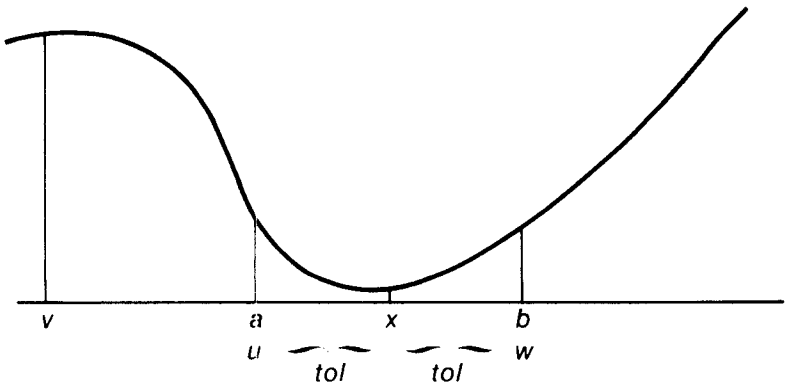


***DIAGRAM 4.2*** **A typical configuration after termination**

## *Section 5*
## *CONVERGENCE PROPERTIES*

There cannot be more than about $2\log_2[(b - a)/tol]$ consecutive parabolic interpolation steps (with the current $a$ and $b$, and the minimum of $tol$ over the interval), for while parabolic interpolation steps are being performed $|p/q|$ decreases by a factor of at least two on every second cycle of the algorithm, and when $|e| \leq tol$ a golden section step is performed. (In this section, "about" means we are not distinguishing between a real number and its integer part. Precise results may easily be obtained as in Section 4.3.) A golden section step does not necessarily decrease $b - a$ significantly, e.g., if $x = b - tol$ and $f(u) < f(x)$, then $b - a$ is only decreased by $tol$, but two golden section steps must decrease $b - a$ by a factor of at least $(1 + \sqrt{5})/2 = 1.618\ldots$. As in Section 4.3, we see that convergence cannot require more than about

$$2K\left[\log_2\left(\frac{b - a}{tol}\right)\right]^2 \tag{5.1}$$

function evaluations, where

$$K = \frac{1}{\log_2[(1 + \sqrt{5})/2]} = 1.44\ldots. \tag{5.2}$$

By comparison, a golden section or Fibonacci search would require about

$$K \log_2\left(\frac{b-a}{tol}\right) \tag{5.3}$$

function evaluations, and a brute-force search about $(b-a)/(2tol)$.

The analogy with procedure *zero* of Chapter 4 should be clear, and essentially the same remarks apply here as were made in Section 4.3. In practical tests convergence has never been more than 5 percent slower than for a Fibonacci search (see Section 6).

In deriving (5.1) we have ignored the effect of rounding errors inside the procedure. As in Section 4.2, it is easy to see that they cannot prevent convergence if floating-point operations satisfy (4.2.10) and (4.2.11), provided the parameter *eps* of procedure *localmin* is at least $2\epsilon$.

### *Superlinear convergence*

If $f$ is $C^2$ near an interior minimum $\mu$ with $f''(\mu) > 0$, then Theorem 3.4.1 shows that convergence is superlinear while rounding errors are negligible. Usually the algorithm stops doing golden section steps, and eventually does only parabolic interpolation steps, with $f(x)$ decreasing at each step, until the tolerance comes into play just before termination. This is certainly true if the successive parabolic interpolation process converges with strong order $\beta_2 = 1.3247\ldots$ (sufficient conditions for this are given in Sections 3.6 and 3.7).

For most of the *ad hoc* methods given in the literature, convergence with a guaranteed error bound of order *tol* in the number of steps given by (5.1) is not certain, and, even if convergence does occur, the order is no greater than for our algorithm. For example, the algorithm of Davies, Swann, and Campey (Box, Davies, and Swann (1969)) evaluates $f$ at two or more points for each parabolic fit, so the order of convergence is at most $\sqrt{\beta_2} = 1.150\ldots$.

## *Section 6*
## *PRACTICAL TESTS*

The ALGOL procedure *localmin* given in Section 8 has been tested using ALGOL W (Wirth and Hoare (1966); Bauer, Becker, and Graham (1968)) on IBM 360/67 and 360/91 computers with machine precision $16^{-13}$. Although it is possible to contrive an example where the bound (5.1) on the number of function evaluations is nearly attained, for our test cases convergence requires, at worst, only 5 percent more function evaluations than are needed to guarantee the same accuracy using Fibonacci search. In most practical

cases superlinear convergence sets in after a few golden section steps, and the procedure is much faster than Fibonacci search.

As an example, in Table 6.1 we give the number of function evaluations required to find the minima of the function

$$f(x) = \sum_{i=1}^{20} \left(\frac{2i - 5}{x - i^2}\right)^2. \tag{6.1}$$

This function has poles at $x = 1^2, 2^2, \ldots, 20^2$. Restricted to the open interval $(i^2, (i + 1)^2)$ for $i = 1, 2, \ldots, 19$, it is unimodal (ignoring rounding errors) with an interior minimum. The fourth column of Table 6.1 gives the number $n_L$ of function evaluations required to find this minimum $\mu_i$, using procedure *localmin* with $eps = 16^{-7}$ and $t = 10^{-10}$ (so the error bound is less than $3tol$, where $tol = 16^{-7}|\mu_i| + 10^{-10}$).

The last column of the table gives the number $n_Z$ of function evaluations required to find the zero of

$$f'(x) = -2\sum_{i=1}^{20} \frac{(2i - 5)^2}{(x - i^2)^3} \tag{6.2}$$

in the interval $[i^2 + 10^{-9}, (i + 1)^2 - 10^{-9}]$, using procedure *zero* (Section 4.6) with $macheps = 16^{-7}$ and $t = 10^{-10}$, so the guaranteed accuracy is nearly the same as for *localmin*. Of course, in practical cases we would seldom be lucky enough to have such a simple analytic expression for $f'$, so procedure

***TABLE 6.1*** **Comparison of procedures *localmin* and *zero***

| $i$ | $\mu_i$ | $f(\mu_i)$ | $n_L$ | $n_Z$ |
|---|---|---|---|---|
| 1 | 3.0229153 | 3.6766990169 | 12 | 14 |
| 2 | 6.6837536 | 1.1118500100 | 11 | 8 |
| 3 | 11.2387017 | 1.2182217637 | 13 | 14 |
| 4 | 19.6760001 | 2.1621103109 | 10 | 12 |
| 5 | 29.8282273 | 3.0322905193 | 11 | 12 |
| 6 | 41.9061162 | 3.7583856477 | 11 | 11 |
| 7 | 55.9535958 | 4.3554103836 | 10 | 11 |
| 8 | 71.9856656 | 4.8482959563 | 10 | 11 |
| 9 | 90.0088685 | 5.2587585400 | 10 | 10 |
| 10 | 110.0265327 | 5.6036524295 | 10 | 10 |
| 11 | 132.0405517 | 5.8956037976 | 10 | 10 |
| 12 | 156.0521144 | 6.1438861542 | 9 | 10 |
| 13 | 182.0620604 | 6.3550764593 | 9 | 10 |
| 14 | 210.0711010 | 6.5333662003 | 9 | 10 |
| 15 | 240.0800483 | 6.6803639849 | 9 | 10 |
| 16 | 272.0902669 | 6.7938538365 | 9 | 10 |
| 17 | 306.1051233 | 6.8634981053 | 9 | 10 |
| 18 | 342.1369454 | 6.8539024631 | 9 | 9 |
| 19 | 380.2687097 | 6.6008470481 | 9 | 9 |

*zero* could not easily be used to find minima of $f$ in this manner. Also, procedure *zero* could find a maximum rather than a minimum.

Table 6.1 shows that the number of function evaluations required by procedure *localmin* compares favorably with the number required by procedure *zero*. Both are much faster than Fibonacci search, which would require 45 function evaluations to find the minimum for $i = 10$ to the same accuracy.

For some numerical results illustrating the superlinear convergence of the successive parabolic interpolation process, see Section 3.9.

## Section 7
## CONCLUSION

The algorithm given in this chapter has the same advantages as the algorithm described in Chapter 4 for finding zeros: convergence in a reasonable number of steps is guaranteed for any function (see equation (5.1)), and on well-behaved functions convergence is superlinear, with order at least 1.3247 . . . , and thus much faster than Fibonacci search. There is no contradiction here: Fibonacci search is the fastest method for the worst possible function, but our algorithm is faster on a large class of functions, including, for example, $C^2$ functions with positive second derivatives at interior minima.

### *A similar algorithm using derivatives*

We pointed out in Section 4.5 that bisection could be combined with interpolation formulas which use both $f$ and $f'$. We could combine golden section search with an interpolation method using both $f$ and $f'$ in a similar way. Davidon (1959) suggests fitting a cubic polynomial to agree with $f$ and $f'$ at two points, and taking a turning point of the cubic as the next approximation. (See also Johnson and Myers (1967).) This method, which gives the possibility of superlinear convergence, could well replace successive parabolic interpolation (using $f$ at three points) in our algorithm if $f'$ is easy to compute. If the cubic has no real turning point, or if the turning point which is a local minimum lies outside the interval known to contain a minimum of $f$, then we can resort to golden section search.

### *Parallel algorithms*

So far we have considered only serial (i.e., sequential) algorithms for finding minima. If a parallel computer is available, more efficient algorithms which take advantage of the parallelism are possible, just as in the analogous zero-finding problem (see Section 4.5). Karp and Miranker (1968) give a parallel search method which is a generalization of Fibonacci search, and

optimal in the same sense, if a sufficiently parallel processor is available. See also Wilde (1964) and Avriel and Wilde (1966). Miranker (1969) gives parallel methods for approximating the root of a function, and these could be used to find a root of $f'$. (Parallel methods for finding a root of $f'$, using only evaluations of $f$, could also be used.) These parallel methods could be combined to give a parallel method with guaranteed convergence, and often superlinear convergence with a higher order than for our serial method.

## *Section 8*
## *AN ALGOL 60 PROCEDURE*

The ALGOL procedure *localmin* for finding a local minimum of a function of one variable is given below. The algorithm and some numerical results are described in Sections 4 to 6. A FORTRAN translation of procedure *localmin* is given in the Appendix.

```
real procedure localmin (a, b, eps, t, f, x);
value a, b, eps, t; real a, b, eps, t, x; real procedure f;
    begin comment:
        If the function f is defined on the interval (a, b), then localmin finds
    an approximation x to the point at which f attains its minimum (or the
    appropriate limit point), and returns the value of f at x. t and eps define
    a tolerance tol = eps|x| + t, and f is never evaluated at two points
    closer together than tol. If f is δ-unimodal (Definition 3.3) for some
    δ < tol, then x approximates the global minimum of f with an error
    less than 3tol (see Section 4). If f is not δ-unimodal on (a, b), then x may
    approximate a local, but non-global, minimum. eps should be no smaller
    than 2macheps, and preferably not much less than sqrt (macheps), where
    macheps is the relative machine precision (Section 4.2). t should be
    positive. For further details, see Section 2.
        The method used is a combination of golden section search and
    successive parabolic interpolation. Convergence is never much slower
    than for a Fibonacci search (see Sections 5 and 6). If f has a continuous
    second derivative which is positive at the minimum (not at a or b) then,
    ignoring rounding errors, convergence is superlinear, and usually the
    order is at least 1.3247 . . . ;
    real c, d, e, m, p, q, r, tol, t2, u, v, w, fu, fv, fw, fx;
    c := 0.381966; comment: c = (3 - sqrt(5))/2;
    v := w := x := a + c × (b - a); e := 0;
    fv := fw := fx := f(x);
    comment: Main loop;
    loop: m := 0.5 × (a + b);
    tol := eps × abs(x) + t; t2 := 2 × tol;
```

```
comment: Check stopping criterion;
if abs(x - m) > t2 - 0.5 × (b - a) then
    begin p: = q: = r: = 0;
    if abs(e) > tol then
        begin comment: Fit parabola;
        r: = (x - w) × (fx - fv); q: = (x - v) × (fx - fw);
        p: = (x - v) × q - (x - w) × r; q: = 2 × (q - r);
        if q > 0 then p: = -p else q: = -q;
        r: = e; e: = d
        end;
    if abs(p) < abs(0.5 × q × r) ∧ p < q × (a - x) ∧
            p < q × (b - x) then
        begin comment: A "parabolic interpolation" step;
        d: = p/q; u: = x + d;
        comment: f must not be evaluated too close to a or b;
        if u - a < t2 ∨ b - u < t2 then d: = if x < m then tol
            else -tol
        end
    else
        begin comment: A "golden section" step;
        e: = (if x < m then b else a) - x; d: = c × e
        end;
    comment: f must not be evaluated too close to x;
    u: = x + (if abs(d) ≥ tol then d else if d > 0 then tol else -tol);
    fu: = f(u);
    comment: Update a, b, v, w, and x;
    if fu ≤ fx then
        begin if u < x then b: = x else a: = x;
        v: = w; fv: = fw; w: = x; fw: = fx; x: = u; fx: = fu
        end
    else
        begin if u < x then a: = u else b: = u;
        if fu ≤ fw ∨ w = x then
            begin v: = w; fv: = fw; w: = u; fw: = fu end
        else if fu ≤ fv ∨ v = x ∨ v = w then
            begin v: = u; fv: = fu
            end
        end;
    go to loop
    end;
localmin: = fx
end localmin;
```