Computers Math. Applic. Vol. 20, No. 2, pp. 23–34, 1990
Printed in Great Britain. All rights reserved

0097-4943/90 $3.00 + 0.00
Copyright © 1990 Pergamon Press plc

A DERIVATIVE-FREE BRACKETING SCHEME FOR UNIVARIATE MINIMIZATION

N. Ghosh[1] and W. W. Hager[2]

[1]Department of Mathematics, The Pennsylvania State University, University Park, PA 16802, U.S.A.

[2]Department of Mathematics, University of Florida, Gainesville, FL 32611, U.S.A.

(*Received* 28 *July* 1989)

Abstract—A derivative-free scheme for univariate minimization is developed. This scheme has a quadratic convergence rate and requires two function evaluations each iteration.

1. INTRODUCTION

This paper develops an algorithm which combines a bracketing strategy, golden section search and a new Newton iteration based on the Lagrange cubic interpolation to obtain the minimum of a function of one variable. These techniques are meshed together in a way that ensures the golden section step used to guarantee convergence of the algorithm will not interfere with the quadratic convergence rate of the underlying Newton iteration. More precisely, we prove that when minimizing a univariate function f, our algorithm converges to a point α which satisfies the second-order necessary conditions: $f'(\alpha) = 0$ and $f''(\alpha) \geqslant 0$. Furthermore, if $f''(\alpha)$ is positive, then the convergence is quadratic. That is, letting e_k denote the error at step k, there exist constants q and r, which are independent of k, such that $e_k \leqslant q/r^{2^k}$ for every k where $r > 1$. Since algorithms to minimize a function of several variables often involve a univariate minimization step, our proposed algorithm can be incorporated in a multivariate scheme.

Let us compare our algorithm to some other derivative-free minimization schemes available in the literature. Of course, the simplest and most fundamental univariate minimization scheme based on the comparison of function values is the Fibonacci search or golden section search (see Refs [1, 2]). Although these schemes are reliable, the convergence is just linear. Schemes utilizing a quadratic fit can be faster than the Fibonacci search or golden section search since the convergence order of the quadratic interpolation iteration is about 1.3 (see Ref. [3, p. 207]). An algorithm that combines quadratic interpolation with the golden section search is developed by Brent [4, Chap. 5]. The fundamental difference between Brent's scheme and our scheme is that Brent's scheme is built around a quadratic interpolation while our scheme is built around a Newton iteration for which the convergence order is two. There are several advantages in our algorithm. Although the quadratic interpolation iteration just requires a few more function evaluations to achieve a given error tolerance than the Newton iteration on a serial computer, on a parallel computer the Newton iteration is more than twice as fast as the quadratic fit scheme since each Newton iteration involves *independent* function evaluations. Moreover, since a convergence order greater than $(1 + \sqrt{5})/2 \approx 1.6$ can never be achieved for a general class of functions by successively minimizing a polynomial that interpolates previous function values (see Ref. [5]), we conclude that for a parallel computer, our Newton iteration is always faster than the successive minimization of interpolating polynomials.

Another advantage in our algorithm is related to numerical stability. In Brent's algorithm, an error tolerance "*eps*" must be provided. If too small a value for *eps* is specified, then the following phenomenon is observed in numerical experiments: initially, one side of the bracketing interval stays fixed while the other side approaches and jumps over the true minimizer. The algorithm then senses that something is wrong and the golden section steps are performed until the "bracketing interval" is small enough. In contrast, our Algorithm 2 has the property that both sides of the bracketing interval typically approach the minimizer simultaneously. If an unrealistic error tolerance is specified, the algorithm performs some golden section steps, however, these steps are applied to a relatively small interval so convergence is rapid. Another numerical advantage related to the accuracy attainable in the Newton iteration is discussed in Section 5. It is important to observe that the point generated by Brent's scheme and our scheme may be different. Our scheme is organized so that it will converge to a local minimizer on the interior of the bracketing interval while Brent's scheme can converge to a local minimizer at an endpoint of the bracketing interval. This distinction is important in the following situation: suppose we wish to compute a minimizer of a univariate function f and it is known that a unique local minimizer lies between a and b. Although f has a unique local minimizer in the interior of the interval $[a, b]$, the restriction of f to $[a, b]$ may have a local minimizer at an endpoint of the interval. Since our algorithm ignores to local minimizer at an endpoint of the interval, the iterations converge to the desired minimizer in the interior of $[a, b]$. Finally, we note that it has not been shown rigorously that the convergence order of Brent's scheme is the same as the convergence order of the underlying quadratic interpolation iteration (see Ref. [6]). For the scheme developed in this paper, we show that the convergence order is the same as the convergence order of the underlying Newton iteration.

In the companion paper [7], a univariate minimization scheme utilizing derivative evaluations and Hermite cubic interpolation is developed. This derivative-based scheme is more numerically stable and obtains an estimate for the minimizer with relative accuracy on the order of the machine epsilon while the scheme developed in this paper obtains relative accuracy on the order of the square root of the machine epsilon. On the other hand, when f can be evaluated faster than its derivative, the derivative-free scheme in this paper is more efficient. For other references to univariate minimization schemes, see Refs [3, 7].

2. THE BRACKETING STRATEGY AND THE FUNDAMENTAL ITERATION

Suppose that f is a real valued function of a real variable. Our univariate minimization scheme assumes that we are given three points a, b and c with b between a and c and with the following property:

$$\begin{array}{llll} f(a) \geqslant f(b) \leqslant f(c), & & \text{if} & a \neq b \neq c, \\ f(a) \geqslant f(b) & \text{and} \quad f'(b)(b-a) \geqslant 0, & \text{if} & a \neq b = c, \\ f(c) \geqslant f(b) & \text{and} \quad f'(b)(b-c) \geqslant 0, & \text{if} & c \neq b = a, \\ f'(b) = 0 & \text{and} \quad f''(b) \geqslant 0, & \text{if} & a = b = c. \end{array}$$

Any triple (a, b, c) with b between a and c and with this property will be called a *bracketing triple*. One strategy for obtaining a bracketing triple is described in Section 2 of Ref. [7]. Given a bracketing triple (a, b, c) and given a point β on the open interval (a, c) with $\beta \neq b$, a new bracketing triple can be constructed using the standard rules (see Ref. [3]):

(R_1) If β lies between a and b and $f(\beta) > f(b)$, then $a^{\text{new}} = \beta$, $b^{\text{new}} = b$ and $c^{\text{new}} = c$.
(R_2) If β lies between a and b and $f(\beta) \leqslant f(b)$, then $a^{\text{new}} = a$, $b^{\text{new}} = \beta$ and $c^{\text{new}} = b$.
(R_3) If β lies between b and c and $f(\beta) \geqslant f(b)$, then $a^{\text{new}} = a$, $b^{\text{new}} = b$ and $c^{\text{new}} = \beta$.
(R_4) If β lies between b and c and $f(\beta) < f(b)$, then $a^{\text{new}} = b$, $b^{\text{new}} = \beta$ and $c^{\text{new}} = c$.

In a similar manner, if $\beta = b$, then after inspecting the sign of $f'(b)$, we can construct a new bracketing triple $(a, b, c)^{\text{new}}$ where the interval $[a, c]$ contains $[a, c]^{\text{new}}$. (Throughout this paper, $[a, c]$ denotes an interval with endpoints a and c. We do not mean to imply that a is less than or equal to c.) When implementing the scheme developed in this paper, we keep a, b, c and β distinct so that a derivative is not needed to update the bracketing triple (see the comments that follow the statement of Algorithm 1 in Section 3). On the other hand, in the theoretical analysis of our scheme, the discussion is simplified if we allow for bracketing triples where two or three points contained in the triple coincide. Observe that if a nested sequence of bracketing triples approaches a limit, then by the mean value theorem, the iteration limit satisfies the second order necessary conditions. More precisely, we have the following.

Lemma 1

Consider a sequence of bracketing triples (a_k, b_k, c_k), where $(a_{k+1}, b_{k+1}, c_{k+1})$ is constructed from (a_k, b_k, c_k) using rules (R_1)–(R_4). If $|a_k - c_k|$ tends to zero and

$$\alpha = \bigcap_{k \geqslant 0} [a_k, c_k],$$

then $f'(\alpha) = 0$ and $f''(\alpha) \geqslant 0$ provided f is twice continuously differentiable near α.

In this paper, the bracketing strategy outlined above is combined with a Newton iteration to obtain a globally quadratically convergent algorithm. To formulate the Newton iteration, we start with three distinct points x, y and z and we introduce a fourth point w defined by $w = x + \sigma gh$, where σ is $+1$ or -1. Let C be the cubic that interpolates f at w, x, y and z. Omitting the algebra, the derivative of C evaluated at x (denoted N) can be expressed:

$$C'(x) = N(x, y, z, w) = \frac{(g-h)f(w) - h_\sigma g^3 f(z) + g_\sigma h^3 f(y) + g_\sigma h_\sigma (g-h)(g_\sigma + h_\sigma - 3) f(x)}{\sigma g h g_\sigma h_\sigma (g-h)},$$

where $h_\sigma = 1 + \sigma h$ and $g_\sigma = 1 + \sigma_g$. Also, the second derivative of C evaluated at x (denoted D) is given by

$$C''(x) = D(x, y, z, w)$$
$$= \frac{(g-h)(g+h)f(w) + \sigma g^3(1-h^2)f(z) + \sigma h^3(g^2-1)f(y) - g_\sigma h_\sigma (g-h)(g+h-\sigma gh)f(x)}{\frac{\sigma}{2} g_\sigma h_\sigma (gh)^2 (g-h)}.$$

In deriving N and D, we restricted x, y, z and w so that no two of these points are the same. With this restriction, the denominator of N and the denominator of D do not vanish. On the other hand, when f has four continuous derivatives, it can be shown that both N and D are well defined even when the denominators vanish since well defined limits exist.

The fundamental iteration that we study in this paper is a variant of the Newton iteration

$$x_{k+1} = x_k - f'(x_k)/f''(x_k).$$

To state our derivative-free implementation of Newton's method, we utilize a map T defined in the following way. Loosely speaking, given a collection of points where f is defined, T extracts out those three points where the value of f is the smallest. More precisely, given a collection of points $z_1, \ldots, z_q$ on which f is defined, let l, m, and n be the three smallest indices with the property that

$$f(z_l) \leqslant f(z_m) \leqslant f(z_n) \leqslant \min\{f(z_i) \colon i = 1, \ldots, q,\ i \neq l,\ m,\ n\},$$

where $l < m$ if $f(z_l) = f(z_m)$ and $m < n$ if $f(z_m) = f(z_n)$. Then $T(z_l, \ldots, z_q)$ denotes the triple (z_l, z_m, z_n). With this notation, iteration k in our derivative-free scheme is essentially the following:

$$w_k = x_k \pm (x_k - y_k)(x_k - z_k),$$
$$v_k = x_k - N(x_k, y_k, z_k, w_k)/D(x_k, y_k, z_k, w_k),$$
$$(x_{k+1}, y_{k+1}, z_{k+1}) = T(x_k, y_k, z_k, v_k, w_k).$$

Our rule for deciding which sign to use in the evaluation of w_k appears in step 2 of Algorithm 1 (see Section 3). To help compare this derivative-free approximation of Newton's method, we prove

Lemma 2

Given three distinct points x, y and z, define $g = x - y$, $h = x - z$ and $w = x \pm gh$. If f has two Lipschitz continuous derivatives on an interval I which contains α in its interior, then

$$\lim_{\substack{x \to \alpha \\ y \to \alpha \\ z \to \alpha}} N(x, y, z, w) = f'(\alpha) \quad \text{and} \quad \lim_{\substack{x \to \alpha \\ y \to \alpha \\ z \to \alpha}} D(x, y, z, w) = f''(\alpha). \tag{1}$$

Moreover, if f is four times continuously differentiable on I and I contains x, y, z and w, then there exists a point $\xi \in I$ such that

$$N(x, y, z, w) = f'(x) \pm \frac{(gh)^2}{24} f^{(4)}(\xi), \tag{2}$$

and if both $|g| \leqslant 1$ and $|h| \leqslant 1$, then there exists a point $\eta \in I$ such that

$$D(x, y, z, w) = f''(x) + \frac{\theta gh}{2} f^{(4)}(\eta), \tag{3}$$

where $|\theta| \leqslant 1$.

Proof. Again, let C be the cubic which interpolates f at the points, x, y, z and w. Since C is equal to f at four points and both C and f are continuously differentiable, Rolle's theorem implies that $C' = f'$ at three points. Similarly, since both C and f are twice continuously differentiable, Rolle's theorem implies that $C' = f'$ at three points. Similarly, since both C and f are twice continuously differentiable, Rolle's theorem implies that $C'' = f''$ at two points. C'' is a linear function that agrees with the Lipschitz continuous function f'' at two points. Therefore, the third derivative of C is bounded by the Lipschitz constant for f''. This bound for the third derivative of C coupled with the fact that $C' = f'$ and $C'' = f''$ somewhere in the convex hull of w, x, y and z yields equations (1).

Now let us consider equation (2). It is well known (see Ref. [8, p. 248]) that the error in cubic interpolation satisfies the relation

$$f(t) - C(t) = (t - w)(t - x)(t - y)(t - z) f[w, x, y, z, t], \tag{4}$$

where $f[w, x, y, z, t]$ is the fourth order divided difference of f based on the points w, x, y, z and t. This fourth order divided difference can be expressed (see [Ref. 8, p. 249]):

$$f[w, x, y, z, t] = \frac{f^{(4)}(\xi)}{24}, \tag{5}$$

where ξ lies in the convex hull of the points, w, x, y, z, t. Differentiating equation (4) with respect to t, evaluating the derivative at $t = x$, and making the substitution (5), we obtain equation (2). It is important to note that just four derivatives are needed for f even though differentiation of equation (5) would appear to require the fifth derivative of f. In fact, as we now show, just one continuous derivative is needed to obtain the identity

$$\lim_{t \to x} (t - x) \frac{\mathrm{d}}{\mathrm{d}t} f[w, x, y, z, t] = 0.$$

Since a divided difference is a symmetric function of its arguments, we have:

$$\begin{aligned}
(t - x) \frac{\mathrm{d}}{\mathrm{d}t} f[w, x, y, z, t] &= (t - x) \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{f[t, w, y, z] - f[w, y, z, x]}{t - x} \right) \\
&= \frac{(t - x) \frac{\mathrm{d}}{\mathrm{d}t} f[w, y, z, t] + f[w, y, z, x] - f[w, y, z, t]}{t - x} \\
&= \frac{\mathrm{d}}{\mathrm{d}t} f[w, y, z, t] - \frac{\mathrm{d}}{\mathrm{d}s} f[w, y, z, s] \bigg|_{s = \xi(t)},
\end{aligned} \tag{6}$$

where $\xi(t)$ lies between t and x. Writing $f[w, y, z, t]$ in terms of the value of f at w, y, z and t, we see that the derivative on the right-hand side of equations (6) can be computed when f is differentiable. Moreover, when this derivative is continuous, the right-hand side of equations (6) approaches zero as t approaches x.

Finally, let us consider equation (3). Without loss of generality, we can assume that $|h| \geqslant |g| \geqslant |gh|$. Since the second derivative of $e(t) = f(t) - C(t)$ vanishes for at least two points, say $t = s_1$ and $t = s_2$, we have the equality (see Ref. [8, p. 249]):

$$e''(x) = (x - s_1)(x - s_2) \frac{f^{(4)}(\eta)}{2},$$

where η lies in the convex hull of s_1, s_2 and x. The relation $|h| \geqslant |g| \geqslant |gh|$ implies that s_1 can be chosen in the convex hull of w, x, and y and s_2 can be chosen in the convex hull of w, x, y, and z. Since $|x - s_1| \leqslant |g|$ and $|x - s_2| \leqslant |h|$, the proof is complete. □

3. IMPLEMENTATION

Our proposed algorithm to compute a local minimum of f has five parts:

Algorithm 1

1. Initialization.
2. Newton step.
3. Test convergence speed.
4. Test cost convexity.
5. Golden section step.

We assume that a bracketing triple (a, b, c) is given. At the start of the iteractions, step 1 is executed. Then in each successive iteration, we perform a Newton step, we test the convergence speed, and we test the convexity of the cost function. If either the convergence is slow or the cost lacks convexity, then a golden section step is performed. Conversely, if the convergence seems fast and the cost appears convex, then a Newton step is performed. The map T_b utilized in step 2 is a slightly modified form of the map T defined in Section 2. In particular, given a collection of points $z_1, \ldots, z_q$ on which f is defined with $z_l = b$ for some l, let m and n be the two smallest indices with the property that $m \neq l \neq n$ and

$$f(z_m) \leqslant f(z_n) \leqslant \min\{f(z_i): i = 1, \ldots, q, \quad i \neq l, m, n\},$$

where $m < n$ if $f(z_n)$. Then $T_b(z_1, \ldots, z_q)$ denotes the triple (b, z_m, z_n). In detail, the five steps of Algorithm 1 are the following.

1. Initialization

Set $(x, y, z) = T(b, a, c)$, define $l = 2|a - c|$, and proceed to step 2.

2. Newton step

Set $w = x \pm (x - y)(x - z)$, where the sign is chosen which yields the value for w closer to the midpoint of the interval $[a, c]$. If $D(x, y, z, w) = 0$, then branch to step 5. Otherwise, set $v = x - N(x,y,z,w)/D(x, y, z, w)$. If $|v - x| > l$ or v is outside the open interval (a, c) or w is both outside (a, c) and $f(w) < f(v)$, then branch to step 5. Otherwise, update the bracketing triple (a, b, c) using rules (R_1)–(R_4) and the following choice for β: If w is outside the interval (a, c), then $\beta = v$ and if w is inside the interval (a, c), then $\beta = v$ or $\beta = w$, whichever yields the smaller value for f. Letting o denote either v or w, whichever yields the larger value for f, update the bracketing triple a second time using $\beta = o$ if o lies inside the new interval (a, c). Finally, perform the update $(x, y, z)^{\text{new}} = T_b(x, y, z, v, w)$ and proceed to step 3.

3. Test convergence speed

If $|y - x| + |z - x| > l$, then branch to step 5. Otherwise, replace the value of l with $l/2$ and proceed to step 4.

4. Test cost convexity

If the second order divided difference $f[x, y, z]$ is negative, then proceed to step 5. Otherwise, branch to step 2.

5. Golden section step

If $x = y$, $N(x, y, z, w) = 0$, and $f''(x) \geqslant 0$, then set $a^{\text{new}} = b^{\text{new}} = c^{\text{new}} = x$ and stop. Otherwise, set $x = b + (a - b)(3 - \sqrt{5})/2$ if $|a - b| \geqslant |b - c|$ and set $x = b + (c - b)(3 - \sqrt{5})/2$ if $|a - b| < |b - c|$. Use rules (R_1)–(R_4) with $\beta = x$ to update the bracketing triple (a, b, c), then branch to step 1.

The map T_b is introduced to help simplify the analysis of Algorithm 1. The analysis is much easier if it is arranged so that in each iteration $x = b$, the middle point in the bracketing triple. Since the point b generated in step 2 is always one of the points x, y, z, v, or w where the value of f is smallest, $T_b(x, y, z, v, w)$ is a triple consisting of three arguments where the value of f is smallest. However, if the function value associated with two or more arguments is identically equal to the minimum of $f(x), f(y), \ldots, f(w)$, then T_b rearranges the three extracted arguments so that b comes first. When implementing Algorithm 1 on a computer, we must guard against the following source for numerical instability: As two arguments of N or D in step 2 approach each other, the relative error in the computed value of N and D increases. On the other hand, small perturbations in x, y, z or w will not effect the convergence speed of the algorithm. Often, when implementing this algorithm, a desired error tolerance t is specified and whenever two arguments of N or D are closer together than t, then their separation is increased to t. In particular, if $|w - x| \leqslant 2t$ in step 2, then we set $w = x \pm t$ where the sign is chosen which yields the value for w closer to the midpoint of the interval $[a, c]$. Likewise, if $|v - x| \leqslant t$, then we set $v = x \pm t$ where the sign is chosen which yields the value for v closer to the midpoint of the interval $[a, c]$. Finally, if $|v - w| \leqslant t$, then we set $v = w \pm t$, where the sign is chosen so that x and v are on opposite sides of w. Typically, the iterations are terminated when $|a - c| \leqslant 2t$. We emphasize that the modifications discussed above are only needed to ensure numerical stability. Theoretically, the Newton step is well defined even if two arguments of N and D are equal (since a limit exists as these arguments approach each other).

When implementing Algorithm 1 on a parallel computer, we simultaneously evaluate f at the point v of the current iteration and at the point w of the next iteration. Although the value of w in the next iteration depends on the relationship between $f(v)$, $f(x)$, $f(y)$ and $f(z)$, there are at most four different ws that need to be considered. Thus one strategy is to simultaneously evaluate f at v and at the four potential ws and then discard the irrelevant function values. Another strategy, which involves no discarded function values, is based on the following observations: the sign convention used in the choice of w is designed so that w lies inside the bracketing interval when the iterations are near a local minimizer (see Lemma 5). On the other hand, the convergence rate is not effected if we always use one sign, say the plus sign, when computing w. Since the inequality $f(v) < f(x)$ typically holds, the value for w in the next iteration is $v + (v - x)(v - y)$ in a large proportion of the cases. Therefore, we can evaluate f at both v and at the anticipated w for the next iteration. If it turns out that the anticipated w is not the actual w, then we must perform another evaluation, however, this extra evaluation is carried out infrequently.

In Algorithm 1, the point w generated in step 2 is a somewhat arbitrary point near x, the best approximation to a local minimizer. It is conceivable that for many iterations, one side of the bracketing triple stays fixed while the other side approaches the local minimizer. But if it can be arranged so that w is close to x while w and x are on opposite sides of the local minimizer, then both sides of the bracketing triple approach the local minimizer as the iterations progress. One strategy to enhance the likelihood that w and x lie on opposite sides of the local minimizer is the following: let q denote the minimizer of the quadratic that interpolates f at x, y, z and define $w = 2q - x$. Algorithm 1 with this new formula for w will be called Algorithm 2. The fact that w and x tend to lie on opposite sides of a local minimizer is connected with subtle relations between error constants associated with the quadratic iteration and with the Newton iteration. A numerical illustration appears in Section 5. To test this property for a wide range of problems, we performed the following experiment: functions of the form $f(x) = ax^2 + bx^3 + cx^4$ were considered where $0.1 \leqslant a \leqslant 100$, $-10 \leqslant b \leqslant 10$ and $-10 \leqslant c \leqslant 10$. Starting from $z_0 = -0.2$, $y_0 = 0.2$, and $x_0 = y_0 - f'(y_0)/f''(y_0)$, up to four iterations of the following algorithm were performed:

$$\begin{aligned} w_k &= 2q_k - x_k, \\ v_k &= x_k - N(x_k, y_k, z_k, w_k)/D(x_k, y_k, z_k, w_k), \\ (x_{k+1}, y_{k+1}, z_{k+1}) &= T(x_k, y_k, z_k, v_k, w_k). \end{aligned}$$

Here q_k denotes the minimizer of the quadratic that interpolates f at x_k, y_k, and z_k. Since f has a local minimum at $x = 0$, the iterations were terminated whenever $|x_k| \leqslant 10^{-16}$. We just considered problems for which f'' did not vanish on the interval $[-0.4, +0.4]$. In over 94% of the iterations, x_k and w_k were on opposite sides of the local minimizer $x = 0$.

For reference, we state the formulas for the minimizer of an interpolating quadratic and for both the first and the second derivative of an interpolating cubic. If f_i denotes $f(x_i)$, then the minimizer of the quadratic that interpolates f at $x = x_0$, $x = x_1$, and $x = x_2$ is

$$q(x_0, x_1, x_2) := x_0 + \frac{1}{2}\left(\frac{(x_1 - x_0)^2(f_0 - f_2) + (x_2 - x_0)^2(f_1 - f_0)}{(x_2 - x_0)(f_1 - f_0) + (x_0 - x_1)(f_2 - f_0)}\right).$$

Defining $d_i = x_i - x_0$, $b_{ij} = d_i d_j(d_i - d_j)$, and $a_{ij} = d_i d_j b_{ij}$, the derivative at $x = x_0$ of the cubic that interpolates f at $x = x_0$, $x = x_1$, $x = x_2$, and $x = x_3$ is

$$N(x_0, x_1, x_2, x_3) := \frac{a_{23}(f_1 - f_0) + a_{31}(f_2 - f_0) + a_{12}(f_3 - f_0)}{d_1 d_2 d_3(b_{23} + b_{31} + b_{12})}. \tag{7}$$

Defining $r_{ij} = d_i d_j(d_i^2 - d_j^2)$ the second derivative at $x = x_0$ of the cubic that interpolates f at $x = x_0$, $x = x_1$, $x = x_2$ and $x = x_3$ is

$$D(x_0, x_1, x_2, x_3) := -\frac{2[r_{23}(f_1 - f_0) + r_{31}(f_2 - f_0) + r_{21}(f_3 - f_0)]}{d_1 d_2 d_3(b_{23} + b_{31} + b_{12})}. \tag{8}$$

Note that the formulas for N and D given in Section 2 are simplified versions of equations (7) and (8) corresponding to a special choice for x_3. With the notation given above, the Newton step of Algorithm 2 can be stated.

2. Newton step of Algorithm 2

If $(z - x)f(y) + (x - y)f(z) + (y - z)f(x) = 0$, then branch to step 5. Otherwise, set $w = 2q(x, y, z) - x$. If $D(x, y, z, w) = 0$, then branch to step 5. Otherwise, set $v = x - N(x, y, z, w)/D(x, y, z, w)$. If $|v - x| > l$ or $|w - x| > l$ or v is outside the open interval (a, c) or w is both outside (a, c) and $f(w) < f(v)$, then branch to step 5. Otherwise, update the bracketing triple (a, b, c) using rules (R_1)–(R_4) and the following choice for β: If w is outside the interval (a, c), then $\beta = v$. Otherwise, $\beta = v$ or $\beta = w$, whichever yields the smaller value for f. Letting o denote either v or w, whichever yields the larger value for f, update the bracketing triple a second time using $\beta = o$ if o lies inside the new interval (a, c). Finally, perform the update $(x, y, z)^{\text{new}} = T_b(x, y, z, v, w)$ and proceed to step 3.

Note that Algorithm 2, unlike Algorithm 1, is not well suited for a parallel computer since the evaluation of f at v and w cannot be performed simultaneously. For a parallel computer, it is better to use Algorithm 1 to generate the iterations. However, at the same time that we evaluate f at the v and w of Algorithm 1, we can also evaluate f at the w of Algorithm 2. By incorporating the w of Algorithm 2 into the update of the bracketing triple, it is possible to push both sides of the bracketing interval toward the local minimizer in each iteration. On a parallel computer, this hybrid algorithm incorporates good features of both Algorithm 1 and Algorithm 2. As with Algorithm 1, numerical errors are relatively large when two arguments of N or D are relatively close together. Again, these arguments can be perturbed slightly to ensure numerical stability. On the other hand, it is much less likely that Algorithm 2 will generate two points close together than Algorithm 1. Near a local minimum, x and w tend to lie on opposite sides of the minimizer and the error in x and w is on the order of the square of the error in either y or z.

4. CONVERGENCE

In this section both the local and the global convergence of Algorithm 1 are analyzed. Each result in this section also applies to Algorithm 2 and the analysis is essentially the same since the distance between w and x in Algorithm 1 is comparable to the distance between w and x in Algorithm 2. In the analysis that follows, a k subscript is attached to a variable to denote its value in iteration k just after v is evaluated in step 2. In stating our results, it is implicitly assumed that f is sufficiently smooth that the operations performed in each iteration are defined. We begin by considering the local convergence of Algorithm 1.

Lemma 3

In each iteration of Algorithm 1, $[a_{k+1}, c_{k+1}] \subset [a_k, c_k]$. Moreover, if an infinite number of golden section steps are performed, then $|a_k - c_k|$ approaches zero as k increases.

Proof. The relation $[a_{k+1}, c_{k+1}] \subset [a_k, c_k]$ follows from the update rules (R_1)–(R_4). Suppose that a golden section step is performed at iteration k. If β_{k+1} is an endpoint of the updated bracketing interval, then we have

$$|a_{k+1} - c_{k+1}| + \frac{\sqrt{5}-1}{2} m_k = |a_k - c_k|,$$

where m_k denotes the maximum of $|a_k - b_k|$ and $|c_k - b_k|$. Since $m_k \geqslant |a_k - c_k|/2$, it follows that

$$|a_{k+1} - c_{k+1}| \leqslant \frac{5-\sqrt{5}}{4} |a_k - c_k|. \tag{9}$$

This shows that the width of the bracketing interval contracts by at least the factor 0.7. Similarly, if $\beta_{k+1} = b_{k+1}$ and β_{k+1} either becomes an endpoint or lies outside of a bracketing interval $[a_{j+1}, c_{j+1}]$ at iteration $j > k$, then we have

$$|a_{j+1} - c_{j+1}| \leqslant \frac{5-\sqrt{5}}{4} |a_k - c_k|.$$

Therefore, if the β_{k+1} generated in a golden section step eventually becomes an endpoint or falls outside of a bracketing interval, then $|a_k - c_k|$ tends to zero as k increases provided an infinite number of golden section steps are performed. Conversely, suppose that an infinite number of golden section steps are performed and there exists an integer K such that $b_k = \beta_K$ for every $k \geqslant K$. If a golden section step is performed at iteration k where $k > K$, then the equality $b_{k+1} = \beta_K$ implies that β_{k+1} is an endpoint of $[a_{k+1}, c_{k+1}]$. By condition (9) we conclude that the width of the bracketing interval contracts by at least the factor 0.7. Again, $|a_k - c_k|$ tends to zero as k increases since the width of the bracketing interval contracts by at least the factor 0.7 whenever the iteration number exceeds K and a golden section step is performed. □

By the structure of step 2, $x_k = b_k$ is contained in the bracketing interval (a_k, c_k) for every k. Lemma 3 tells us that the width of the bracketing interval shrinks to zero if an infinite number of golden section steps are performed. Consequently, when an infinite number of golden section steps are performed, the x_k approach a limit α. By Lemma 1, $f'(\alpha) = 0$ and $f''(\alpha) \geqslant 0$ provided f is twice continuously differentiable near α. Conversely, suppose that a finite number of golden section steps are performed. Since the l_k approach zero at least as fast as a constant time 2^{-k} and since x_k is an element of the set $\{x_{k+1}, y_{k+1}, z_{k+1}\}$ for every k, it follows from step 3 that x_k, y_k, and z_k all approach a limit α. Since a finite number of golden section steps are performed, the convexity test is passed for k sufficiently large. Since the second order divided difference $f[x, y, z]$ is equal to the second derivative of f somewhere in the convex hull of x, y, and z, it follows that $f''(\alpha) \geqslant 0$. Moreover, the inequality $|v_k - x_k| \leqslant l_k$ contained in step 2 implies that $N(x_k, y_k, z_k, w_k)/D(x_k, y_k, z_k, w_k)$ approaches zero as k increases. Under the hypotheses of Lemma 2, $f'(\alpha) = 0$. These observations are summarized in the following.

Theorem 1

The x_k generated by Algorithm 1 approach a limit α contained in the initial bracketing interval $[a_0, c_0]$. If f has two Lipschitz continuous derivatives in a neighborhood of α, then $f'(\alpha) = 0$ and $f''(\alpha) \geqslant 0$.

Now let us suppose that the x_k generated by Algorithm 1 approach a limit α for which $f'(\alpha) = 0$ and $f''(\alpha) > 0$ and let us analyze the convergence rate. We will show that if f is four times continuously differentiable in a neighborhood of α, then the golden section step is not invoked for k sufficiently large and the root convergence order of the iterations is at least two. We begin with two lemmas which will show that the convergence is "fast" and both v_k and w_k are inside the interval (a_k, c_k) for k sufficiently large.

Lemma 4

If f is four times continuously differentiable in a neighborhood of a local minimizer α of f and $f''(\alpha)$ is positive, then there exists a neighborhood N of α and there exists a constant ρ such that

$$|v-\alpha| \leqslant \rho(|x-\alpha|+|x-y||x-z|)^2, \tag{10}$$

for every x, y and z in N where v is generated by step 2 of Algorithm 1.

Proof. Expanding $f'(\alpha)$ in a Taylor series about x, we have

$$0=f'(\alpha)=f'(x)-ef''(x)+\frac{e^2}{2}f^{(3)}(\xi),$$

where $e=x-\alpha$ and ξ lies between α and x. Solving for $f'(x)$ yields

$$f'(x)=ef''(x)-\frac{e^2}{2}f^{(3)}(\xi). \tag{11}$$

Subtracting α from each side of the equation $v=x-N(x,y,z,w)/D(x,y,z,w)$ and utilizing both Lemma 2 and equation (11) gives us

$$v-\alpha=e-\frac{ef''(x)+O(e^2)+O(p^2)}{f''(x)+O(|p|)},$$

where $p=(x-y)(x-z)$. Since $f''(\alpha)>0$, the conclusion of the lemma follows almost directly. □

Lemma 5

If the x_k generated by Algorithm 1 converge to a local minimizer α and $f''(\alpha)$ is positive, then there exists a neighborhood N of α with the property that whenever x_k, y_k and z_k are contained in N and $a_k \neq c_k$, then either the v_k and the w_k generated by step 2 are contained in the open interval (a_k, c_k) or $a_{k+1}=b_{k+1}=c_{k+1}=\alpha$.

Proof. Let us consider a neighborhood of α where f'' is positive. Since f' is monotone in this neighborhood, $f(y) \geqslant f(x)$ when $y \geqslant x \geqslant \alpha$. Similarly, $f(y) \geqslant f(x)$ when $y \leqslant x \leqslant \alpha$. It is now demonstrated that in each iteration, we either have $a_k=y_k$ and $f(z_k) \leqslant f(c_k)$ or we have $c_k=y_k$ and $f(z_k) \leqslant f(a_k)$. We observed earlier that $b_k=x_k$ for every k. By the structure of step 2, y_k and z_k cannot lie on the open interval (a_k, c_k). If neither y_k nor z_k is an endpoint of the interval $[a_k, c_k]$, then the monotonicity of f' implies that both $f(y_k)$ and $f(z_k)$ are larger than the minimum of $f(a_k)$ and $f(c_k)$. But this violates the requirement that x_k, y_k, and z_k are the previously computed points with smallest value. Similarly, if $a_k=y_k$ and $f(z_k)>f(c_k)$, then we again violate the requirement that x_k, y_k, and z_k are the previously computed points with the smallest value.

Let us now assume that $a_k=y_k$ and $f(z_k) \leqslant f(c_k)$ and let us consider a neighborhood of α where $|x_k-z_k| \leqslant 1/2$. Since $x_k=b_k$ for every k, we have $|x_k-y_k||x_k-z_k| \leqslant |x_k-y_k|/2 = |b_k-a_k|/2 \leqslant |c_k-a_k|/2$. Since $|x_k-w_k|=|x_k-y_k||x_k-z_k| \leqslant |c_k-a_k|/2$, it follows from step 2 that w_k lies between a_k and c_k when the iterations lie in a neighborhood of α. Now consider v_k. If a finite number of golden section steps are performed, then v_k lies on the open interval (a_k, c_k) for k sufficiently large and we are done. If an infinite number of golden section steps are performed, then there exists an integer K such that for $k \geqslant K$, x_k, y_k, z_k, a_k and c_k are all contained in a neighborhood of α where both condition (10) and the hypotheses of Ref. [7, Lemma 4] are satisfied. Henceforth, it is assumed that $k \geqslant K$. We will show that in a neighborhood of α, $|v_k-\alpha| \leqslant$ the minimum of $|a_k-\alpha|$ and $|c_k-\alpha|$ so that v_k must lie between a_k and c_k. The inequality $f(a_k) \geqslant f(b_k)=f(x_k)$ and Ref. [7, Lemma 4] yield the estimate $|x_k-\alpha| \leqslant 2|a_k-\alpha|$. Lemma 4 tells us that $|v_k-\alpha| \leqslant \rho(e_k+p_k)^2$ where $e_k=|x_k-\alpha|$ and $p_k=|x_k-y_k||x_k-z_k|$. If $e_k<p_k$, it follows that $|v_k-\alpha| \leqslant 4\rho p_k^2$. The inequality $f(x_k) \leqslant f(y_k)$ and Ref. [7, Lemma 4] imply that $|x_k-y_k| \leqslant e_k+|y_k-\alpha| \leqslant 3|a_k-\alpha|$. Likewise, the inequalities $f(x_k) \leqslant f(c_k)$ and $f(z_k) \leqslant f(c_k)$ imply that $|x_k-z_k| \leqslant |x_k-\alpha|+|z_k-\alpha| \leqslant 4|c_k-\alpha|$. Hence, we have $|v_k-\alpha| \leqslant 4\rho p_k^2 \leqslant 576\rho|a_k-\alpha|^2|c_k-\alpha|^2 \leqslant 1/2$ times the minimum of $|a_k-\alpha|$ and $|c_k-\alpha|$ in neighborhood of α. Conversely, if $e_k \geqslant p_k$, then it follows from Lemma 4 that $|v_k-\alpha| \leqslant 4\rho p_k^2$. Consequently, in a neighborhood of α, $|v_k-\alpha| \leqslant e_k/4$. By Ref. [7, Lemma 4], we have $|a_k-\alpha| \geqslant e_k/2$ and

$|c_k - \alpha| \geqslant e_k/2$. If neither a_k nor c_k is equal to α, then the inequality $|v_k - \alpha| \leqslant e_k/4$ implies that v_k must lie on the open interval (a_k, c_k). If $c_k = \alpha$, then $a_k = b_k = c_k = \alpha$ since $f(c_k) \geqslant f(z_k) \geqslant f(y_k) = f(a_k) \geqslant f(b_k)$. But this is impossible since $a_k \neq c_k$. If $a_k = \alpha$, then $x_k = y_k = v_k = \alpha$ and in the first part of the golden section step, we set $a_{k+1} = b_{k+1} = c_{k+1} = \alpha$. This completes the proof. □

Theorem 2

If the x_k generated by Algorithm 1 convergence to a local minimizer α and $f''(\alpha)$ is positive, then there exist constants q and r, independent of k, such that $|x_k - \alpha| \leqslant q/r^{2k}$ for every k where $r > 1$.

Proof. Since $f''(\alpha) > 0$, the convexity test is always passed in a neighborhood of α. By Lemma 5, v_k and w_k are contained on the interval (a_k, c_k) when the iterations are sufficiently close to α. Thus for k sufficiently large, a golden section step is only performed when the convergence is "slow". We now use Lemma 4 to show that the convergence is quadratic. Throughout the proof, C denotes a generic constant which is independent of k, but which may have different values in different equations. Since the qualifier "in a neighborhood of α" applies to almost every inequality in this proof, we sometimes omit the qualifier to minimize repetition. By Lemma 4, $|v_k - \alpha| \leqslant C(e_k + p_k)^2$ were $p_k = |x_k - y_k|\,|x_k - z_k|$. Let u_k denote $|z_k - \alpha|$. Applying the triangle inequality, we have

$$p_k \leqslant (e_k + |y_k - \alpha|)(e_k + u_k). \tag{12}$$

By Ref. [7, Lemma 4], $e_k \leqslant 2u_k$ and $|y_k - \alpha| \leqslant 2u_k$ in a neighborhood of α. Hence, condition (12) gives us the relation $p_k \leqslant 12u_k^2$ and by Lemma 4, we have $|v_k - \alpha| \leqslant C(e_k + u_k^2)^2$. Referring to Algorithm 1, observe that the convergence speed test in step 3 is always passed the first iteration after a golden section step since both v_k and w_k lie on the interval $[a_k, c_k]$. Let us suppose that $(x_{k+1}, y_{k+1}, z_{k+1}) = T_b(x_k, y_k, z_k, v_k, w_k)$. Since $f(x_{k+1}) \leqslant f(v_k)$, Ref. [7, Lemma 4] also tells us that $e_{k+1} \leqslant 2|v_k - \alpha|$. Therefore, we have the relation

$$e_{k+1} \leqslant C(e_k + u_k^2)^2. \tag{13}$$

Since $f(z_{k+1}) \leqslant$ the largest of $f(x_k)$, $f(v_k)$, and $f(w_k)$, Ref. [7, Lemma 4] implies that $u_{k+1} \leqslant 2e_k + 2|v_k - \alpha| + 2|w_k - \alpha|$. The inequality $|w_k - \alpha| \leqslant |x_k - \alpha| + |x_k - w_k| = e_k + p_k$ coupled with the previous estimates for p_k and $|v_k - \alpha|$ tell us that

$$u_{k+1} \leqslant C(e_k + u_k^2). \tag{14}$$

Adding the square of condition (14) to condition (13) gives

$$e_{k+1} + u_{k+1}^2 \leqslant C(e_k + u_k^2)^2. \tag{15}$$

Thus the quantity $e_k + u_k^2$ converges to zero quadratically. In particular, the error e_k is bound by an expression of the form q/r^{2k}.

To complete the proof, we show that in a neighborhood of α, the convergence is fast enough that a golden section step is not performed due to slow convergence. There are two places where the algorithm can jump to a golden section step due to slow convergence: steps 2 and 3. In step 3 we monitor the quantity $|x_k - y_k| + |x_k - z_k|$. By the triangle inequality, $|x_k - y_k| + |x_k - z_k| \leqslant 2e_k + u_k + |y_k - \alpha|$. Since $f(y_k) \leqslant f(z_k)$, Ref. [7, Lemma 4] implies that $|y_k - \alpha| \leqslant 2u_k$ and $|x_k - y_k| + |x_k - z_k| \leqslant 2e_k + 3u_k$. From inequality (15), we conclude that the quantity $|x_k - y_k| + |x_k - z_k|$ converges to zero much faster than a constant times 2^{-k}. More precisely, if an initialization step is performed at iteration j, then for a neighborhood of α, we have $|x_k - y_k| + |x_k - z_k| \leqslant l_j/2^{k-j-1}$ for each $k > j$. Furthermore, the quantity $2^k(|x_k - y_k| + |x_k - z_k|)$ tends to zero as k increases. In step 2 the difference $|v_k - x_k|$ is monitored. During the proof of Lemma 4, we obtained the estimate

$$|v_k - x_k| = \left|\frac{e_k f''(x_k) + O(e_k^2) + O(p_k^2)}{f''(x_k) + O(p_k)}\right|.$$

Again, the bounds established for e_k and p_k tell us that $|v_k - x_k|$ converges to zero much faster than a constant times 2^{-k}. In summary, the iterations will not jump to a golden section step due to slow convergence and the proof is complete. □

Table 1. Minimizing function (16) using Algorithm 1

k	x_k	w_k	a_k	c_k
0	1.10000000000	1.10700000000	0.80000000000	1.20000000000
1	1.01513728324	1.01048148404	0.80000000000	1.07000000000
2	1.00029516203	1.00014397540	0.80000000000	1.01048148404
3	1.00000009863	1.00000005618	0.80000000000	1.00014397540
4	1.00000000000	1.00000000000	0.80000000000	1.00000005618
5	1.00000000000	1.00000000000	0.80000000000	1.00000000000

Table 2. Minimizing function (16) using Algorithm 2

k	x_k	w_k	a_k	c_k
0	1.10000000000	0.86521739130	0.80000000000	1.20000000000
1	1.01026222078	0.97624406339	0.86521739130	1.10000000000
2	1.00005291611	0.99970269959	0.97624406339	1.01026222078
3	0.99999997426	1.00000001002	0.99970269959	1.00005291611
4	1.00000000000	1.00000000000	0.99999997426	1.00000001002
5	1.00000000000	1.00000000000	1.00000000000	1.00000000000

5. CONCLUDING REMARKS

One reason that our Newton iteration is preferable to an iteration that interpolates f at the previous x_k is that the Newton iteration often yields a more accurate approximation to the minimizer α. Recall Ref. [9] that the machine epsilon for a computer is the smallest number ϵ with the property that $1 + \epsilon > 1$ when the sum $1 + \epsilon$ is evaluated using machine arithmetic. If α is a local minimizer for f and $f''(\alpha) > 0$, then using machine arithmetic, we typically find that $f(x) = f(\alpha)$ when $|x - \alpha|$ is on the order of $\sqrt{\epsilon}$. Therefore, α can only be determined numerically with accuracy on the order of $\sqrt{\epsilon}$.

Let x_{k+1} denote the last iteration computed by some algorithm and suppose that the error in x_{k+1} is $O(\epsilon^{1/2})$. If the convergence order is p, then x_k typically has error $O(\epsilon^{1/2p})$ and $f(x_k)$ differs from the minimum value $f(\alpha)$ by $O(\epsilon^{1/p})$—the error in $f(x_k)$ is on the order of the square of the error in x_k since $f'(\alpha) = 0$. For simplicity, let us suppose that $f(\alpha) = 0$ so that $f(x_k) = O(\epsilon^{1/p})$. Assuming roundoff errors in the evaluation of f are on the order of ϵ, the computed value of f at x_k is on the order of ϵ and the relative error in the computed value of f at x_k is $O(\epsilon^{1-1/p})$. Since x_{k+1} is evaluated using the computed value of f at x_k, the computed x_{k+1} is generally less accurate than the computed $f(x_k)$. In others words, the relative error in x_{k+1} is at least $O(\epsilon^{1-1/p})$. If $p < 2$, then the iteration never attains the "optimal" $O(\epsilon^{1/2})$ accuracy. The optimal accuracy is only achieved when $p \geqslant 2$. (Recall [10] that a convergence order $\geqslant 2$ can never be achieved using function values at the previous x_k).

To compare the iterations generated by the Newton step of Algorithm 1 to the corresponding Newton step of Algorithm 2, let us numerically evaluate the minimizer $x = 1$ of the function

$$f(x) = x^4 - 3x^3 + 4x^2 - 3x + 1. \tag{16}$$

Table 1 presents the iterations generated by Algorithm 1 while Table 2 presents the iterations generated by Algorithm 2. Observe that x_k and w_k in Table 2 are on opposite sides of the local minimizer $\alpha = 1$ and the bracketing interval shrinks to zero while just one side of the bracketing interval approaches the minimizer in Table 1.

Acknowledgement—The second author (W.W.H.) was supported by National Science Foundation Grant DMS-8520926.

REFERENCES

1. S. M. Johnson, Best exploration for maximum is Fibonaccian. Report RM-1590, RAND Corporation, Santa Monica, Calif. (1955).
2. J. Kiefer, Sequential minimax search for a maximum. *Proc. Am. math. Soc.* **4**, 503–506 (1953).
3. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass. (1984).
4. R. P. Brent, *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, N.J. (1973).
5. A. Tamir, Rates of convergence of a one-dimensional search based on interpolating polynomials. *J. Optimiz. Theory Applic.* **27**, 187–203 (1979).
6. S. M. Robinson, Quadratic interpolation is risky. *SIAM Jl numer. Analysis* **16**, 377–379 (1979).

7. W. W. Hager, A derivative-based bracketing scheme for univariate minimization and the conjugate gradient method. *Computers Math. Applic.* **18**, 779–795 (1989).
8. E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. Wiley, New York (1966).
9. G. E. Forsythe, M. A. Malcom and C. B. Moler, *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs, N.J. (1977).
10. J. F. Traub, *Iterative Methods for the Solution of Equations*. Prentice-Hall, Englewood Cliffs, N.J. (1964).