Computers Math. Applic. Vol. 18, No. 9, pp. 779–795, 1989
Printed in Great Britain.

0097-4943/89 $3.00 + 0.00

A DERIVATIVE-BASED BRACKETING SCHEME FOR UNIVARIATE MINIMIZATION AND THE CONJUGATE GRADIENT METHOD

W. W. HAGER†
Department of Mathematics, University of Florida, Gainesville, FL 32611, U.S.A.

(*Received* 19 *October* 1988)

Abstract—A derivative-based scheme for univariate minimization is developed. This scheme has a quadratic convergence rate and requires an evaluation of the function and its derivative in each iteration. An application involving the conjugate gradient method is presented.

1. INTRODUCTION

This paper develops a univariate minimization algorithm that we refer to as the Cubic Algorithm. This algorithm combines a bracketing strategy, the bisection method, and a Newton iteration based on Hermite cubic interpolation to minimize a function of one variable. These techniques are meshed together in a way that ensures the bisection strategy 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 with $r > 1$ such that $e_k \leqslant q/r^{2^k}$ for every k. 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. In Section 6 we show that the Cubic Algorithm can be combined with the conjugate gradient method in a way that preserves the fundamental convergence properties of the conjugate gradient method; in particular, an efficient technique to preserve both guaranteed convergence and n-step quadratic convergence of the conjugate gradient method is presented. The paper concludes with a series of numerical experiments that compare our Conjugate Gradient Search Scheme to various other algorithms for unconstrained optimization. The experimental results are very encouraging.

We conclude this section with a brief review of literature concerning cubic-based univariate minimization algorithms. Some classic optimization books such as [1] by Pierre and [2] by Walsh formulate a cubic interpolation iteration. In some of these schemes it is possible that one side of the interval bracketing a local minimizer remains fixed while the other side creeps toward the local minimizer at a linear rate. Mifflin [3] develops a derivative-based algorithm with rules similar to our rules for updating the bracketing interval. Mifflin's algorithm, however, is tailored to nonsmooth Lipschitz continuous functions and the convergence is at best superlinear. Dennis and Schnabel [4] develop fast cubic interpolation schemes for finding a point that satisfies an Armijo condition. Although this objective is not the same as searching for a local minimum, the point which satisfies the Armijo condition may be a good approximation to a local minimum. More recently, Al-Baali and Fletcher [5] present some efficient line search routines that generate a point which satisfies a Goldstein–Wolf–Powell convergence criterion. Again, [5] is mainly concerned with efficient low accuracy searches for nonlinear least squares while our paper focuses on the problem of computing a local minimum. The report [6] by Gill and Murray provides an insightful discussion of various issues that must be considered when designing an efficient, robust univariate minimization routine.

†This research was supported by National Science Foundation Grants MCS-8101892 and DMS-8401758 and by Air Force Office of Scientific Research Grant AFOSR-ISSA-860091.

In the companion paper [7], a derivative-free univariate minimization scheme is developed. The scheme in [7] achieves quadratic convergence using two function evaluations each iteration while the scheme in this paper achieves quadratic convergence using one function and one derivative evaluation each iteration. 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 derivative-free scheme obtains relative accuracy on the order of the square root of the machine epsilon. In applications where derivatives are readily available and the derivative can be computed almost as quickly as the function value, the algorithm in this paper will be preferable to the algorithm in [7]. Additional references to univariate minimization schemes are found in [7] and [8].

2. THE BRACKETING STRATEGY

Suppose that f is a real valued differentiable function of a real variable. Our algorithm generates a sequence of nested intervals with the following property (see [9]): If $[a, b]$ denotes a typical interval in the sequence, then

$$f'(a)(b-a) \leqslant 0 \quad \text{and} \quad f(b) \geqslant f(a). \tag{1}$$

Throughout this paper, $[a, b]$ denotes an interval with endpoints a and b. We do not mean to imply that a is less than or equal to b. Now given A and B for which $f(B) \geqslant f(A) \leqslant f(0)$ and A is between B and 0, the following rules produce an interval $[a, b]$ which satisfies (1):

(R_1) If $f'(A)A \geqslant 0$, then $a = A$ and $b = 0$.

(R_2) If $f'(A)A < 0$, then $a = A$ and $b = B$.

If $C \neq 0$ is an approximation to a local minimizer of f and $f'(0)C < 0$, then one approach for finding A and B such that $f(B) \geqslant f(A) \leqslant f(0)$ is the following:

(i) If $f(C) \geqslant f(0)$, then evaluate $f(C_k)$ for $k = 1, 2, \ldots$ where $C_k = \rho^{-k}C$ and $\rho > 1$. Stop evaluation process at the first k for which $f(C_k) < f(0)$ and set $A = C_k$ and $B = C_{k-1}$. Since C and $f'(0)$ have opposite signs, $f(C_k) < f(0)$ for k sufficiently large.

(ii) If $f(C) < f(0)$, then evaluate $f(C_k)$ for $k = 1, 2, \ldots$ where $C_k = \rho^k C$ and $\rho > 1$. Stop evaluation process when $f(C_{k+1}) \geqslant f(C_k)$ and set $A = C_k$ and $B = C_{k+1}$. Provided $f(C_k)$ is not a strictly decreasing function of k, the inequality $f(C_{k+1}) \geqslant f(C_k)$ holds for some k.

In the case $f(C) \geqslant f(0)$, Dennis and Schnabel's cubic backtracking strategy [4, Section 6.3.2] is an alternative to (i):

Using rules (R_1) and (R_2), we can construct an interval which satisfies (1). Now given an interval $[a_k, b_k]$ which satisfies (1) and given a point c_k between a_k and b_k, the following rules produce a subinterval $[a_{k+1}, b_{k+1}]$ which also satisfies (1):

(R_3) If $f(c_k) > f(a_k)$, then $a_{k+1} = a_k$ and $b_{k+1} = c_k$.

(R_4) If $f(c_k) < f(a_k)$, then $a_{k+1} = c_k$ and $b_{k+1} = a_k$ if $f'(c_k)(a_k - c_k) \leqslant 0$

while $a_{k+1} = c_k$ and $b_{k+1} = b_k$ if $f'(c_k)(a_k - c_k) > 0$.

(R_5) If $f(c_k) = f(a_k)$, then $a_{k+1} = c_k$ and $b_{k+1} = a_k$ if $f'(c_k)(a_k - c_k) < 0$,

$a_{k+1} = a_k$ and $b_{k+1} = c_k$ if both $f'(c_k)(a_k - c_k) \geqslant 0$ and $f'(a_k)(b_k - a_k) < 0$, and

$a_{k+1} = c_k$ and $b_{k+1} = b_k$ if both $f'(c_k)(a_k - c_k) \geqslant 0$ and $f'(a_k)(b_k - a_k) \geqslant 0$.

Although rules (R_3) and (R_4) are consistent with the update procedure of [9], the way we treat the case $f(c_k) = f(a_k)$ differs from [9]. Observe that if a bracketing interval $[a, b]$ satisfies (1) and either $f(b) > f(a)$ or $f'(a)(b-a) < 0$, then the cubic which is tangent to f at $x = a$ and at $x = b$ has a unique local minimizer on the interval $[a, b]$. Rule (R_5) is designed so that if the bracketing interval $[a_k, b_k]$ satisfies (1) and either $f(b_k) > f(a_k)$ or $f'(a_k)(b_k - a_k) < 0$, then the updated interval $[a_{k+1}, b_{k+1}]$ satisfies (1) and either $f(b_{k+1}) > f(a_{k+1})$ or $f'(a_{k+1})(b_{k+1} - a_{k+1}) < 0$. Note that the A generated by (i) or (ii) has the property that $f(B) \geqslant f(A) < f(0)$, and in this case, rules (R_1) and (R_2) generate an interval $[a, b]$ for which either $f(b) > f(a)$ or $f'(a)(b-a) < 0$.

If we start with an interval which satisfies (1) and if we construct nested subintervals using rules (R_3)–(R_5), then each subinterval satisfies (1). Furthermore, when these subintervals approach a limit α, the second-order necessary conditions are satisfied at α:

Lemma 1

Suppose that the intervals $[a_k, b_k]$ are updated using rules (R_3)–(R_5) where $[a_0, b_0]$ satisfies (1) and c_k lies between a_k and b_k for each k. If the width of the intervals tends to zero and

$$\alpha = \bigcap_{k \geq 0} [a_k, b_k],$$

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

Proof. Choose k large enough that the interval $[a_k, b_k]$ is contained in the neighborhood of α where f is twice continuously differentiable. Since $f(b_k) \geq f(a_k)$, the mean value theorem implies that there exists ρ_k between a_k and b_k such that $f'(\rho_k)(b_k - a_k) \geq 0$. Since $f'(a_k)(b_k - a_k) \leq 0$, f' vanishes at some point ξ_k between a_k and ρ_k. Also, by the mean value theorem, there exists η_k between a_k and ρ_k such that $f''(\eta_k) \geq 0$. Since ξ_k and η_k approach α as k increases, we conclude that $f'(\alpha) = 0$ and $f''(\alpha) \geq 0$. □

3. CUBIC APPROXIMATIONS

The simplest choice for c_k is the midpoint of the interval $[a_k, b_k]$. With this choice, $[a_{k+1}, b_{k+1}]$ is half as wide as $[a_k, b_k]$ and the diameter of $[a_k, b_k]$ is 2^{-k} times the diameter of $[a_0, b_0]$. This simple choice, however, does not utilize the known value and the known derivative of f. A faster scheme will be developed which uses Hermite cubic interpolation. The local minimizer of the cubic that interpolates the value and the derivative of f at $x = a$ and $x = b$ can be computed using Davidon formula [10, equation 5.1]:

$$c = b - \Delta \frac{f'(b) + w - v}{f'(b) - f'(a) + 2w}, \tag{2}$$

where $\Delta = b - a$ and v and w are given by

$$v = f'(a) + f'(b) - 3(f(b) - f(a))/\Delta \quad \text{and} \quad w = \operatorname{sgn}(\Delta)[v^2 - f'(a)f'(b)]^{1/2}. \tag{3}$$

In (3) "sgn" denotes the sign function defined by $\operatorname{sgn}(\Delta) = +1$ if $\Delta > 0$ and $\operatorname{sgn}(\Delta) = -1$ if $\Delta < 0$. Davidon's paper [10] considers an interval $[a, b]$ where $b > a$ so the "sgn" factor does not appear in his formula for the cubic's minimizer. As a natural extension of (2), Davidon's scheme was later employed in an iterative fashion. That is, a sequence $x_1, x_2, x_3, \ldots$ was computed where x_{k+1} was determined from x_k and x_{k-1} using (2) with a, b, and c identified with x_{k-1}, x_k, and x_{k+1} respectively. Unfortunately, in some of these iterative schemes, the "sgn" factor was omitted. This omission can be devastating since an improper sign produces the maximizer of the cubic, not the minimizer. This omission still appears in at least one book published in 1984.

Davidon's formula (2) for the local minimizer yields relatively high computational accuracy when (1) is satisfied. We now present an alternative high accuracy formula for the minimizer of an interpolating cubic:

$$c = a + \Delta \frac{f'(a)}{f'(a) + v - w} \quad \text{or} \quad c = b - \Delta \frac{f'(b)}{f'(b) + v + w}. \tag{4}$$

The expressions in (4) are derived in the Appendix. Generally, one of the expressions in (4) is more accurate than the other expression. For example, if b is near a local maximizer of the Hermite cubic interpolant, then the second formula in (4) must be avoided. When b is the local maximizer of the Hermite cubic interpolant, $f'(b) = 0$ and the ratio

$$\frac{f'(b)}{f'(b) + v + w} \tag{5}$$

reduces to zero over zero (if this ratio is not zero over zero, then by (4) the minimizer of the cubic is b, which is impossible since b is the maximizer). Theoretically, the ratio (5) approaches a limit

as b approaches the cubic's maximizer. Computationally, the relative error in the computed value of the numerator and the computed value of the denominator approaches infinity as b approaches the local maximizer. Thus the computed ratio (5) is relatively inaccurate when b approaches the local maximizer while the ratio is zero over zero at the local maximizer. If b is near a local maximizer of the Hermite cubic interpolant, then the first formula in (4) can be utilized.

One way to implement (4) is to always utilize the expression for which the denominator is largest in absolute value. Later we observe (see Lemma 3) that the cubic p that is tangent to f at $x = a$ and at $x = b$ has the following second derivatives:

$$p''(a) = -2(f'(a) + v)/\Delta \tag{6}$$

and

$$p''(b) = 2(f'(b) + v)/\Delta. \tag{7}$$

In the Appendix, we see that either w is pure imaginary and p is monotone with no minimum or $w/\Delta \geqslant 0$. In the case where p has a local minimum, both denominators in (4) vanish if and only if $p''(a) = p''(b) = -2w/\Delta \leqslant 0$. Since p'' is linear, we conclude that p'' is a nonpositive constant. If $p'' < 0$, then p has no local minimizer. If $p'' = 0$, then p is linear—a linear function has either no minimum or an infinite number of minimizers when the line is flat. In summary, if p has a local minimizer and p is not constant, then at least one of the denominators in (4) does not vanish.

Observe that if $\Delta(f'(a) + v) < 0$ and w is real, then $p''(a) > 0$ and the first denominator in (4) is negative. If $\Delta(f'(b) + v) > 0$ and w is real, then $p''(b) > 0$ and the second denominator in (4) is positive. If (1) holds, then either $p''(a) \geqslant 0$ or $p''(b) \geqslant 0$. Furthermore, we have

Lemma 2

If (1) holds and at least one of the inequalities in (1) is strict, then either $\Delta(f'(a) + v) < 0$ or $\Delta(f'(b) + v) > 0$.

Proof. Since p is cubic, p'' is linear. In the proof of Lemma 1, we show that if (1) holds, then there exists η between a and b such that $p''(\eta) \geqslant 0$. Referring to the proof of Lemma 1, this inequality is strict whenever $f(b) > f(a)$ or $f'(a_k)(b_k - a_k) < 0$. Since $p''(x)$ is a linear function of x which is positive at $x = \eta \in [a, b]$, $p''(x)$ must be positive either at $x = a$ or at $x = b$. By (6) and (7), we conclude that either $\Delta(f'(a) + v) < 0$ or $\Delta(f'(b) + v) > 0$. □

As we will soon see, the expressions in (4) are more convenient than (2) for the analysis of the cubic iteration. To compare the numerical accuracy of (2) with (4), let us consider the cubic $f(x) = x^2(1 - x)$ which has a local minimum at $x = 0$. For each choice of a and b, the c given by either (2) or (4) should be zero. Thus the numerical error in the computed c is its absolute value. In Table 1, we sum the numerical errors corresponding to various choices of a and b using a VAX 780 computer and single precision arithmetic. In the case where both $p''(a) > 0$ and $p''(b) > 0$, we use the first expression in (4) when $|f'(a)| \leqslant |f'(b)|$ and we use the second expression when $|f'(a)| > |f'(b)|$.

As these experiments (and others) tend to indicate, (2) and (4) have comparable accuracy near a local minimizer while (4) is more accurate than (2) when one of the evaluation points is in a region where the second derivative is positive while the other evaluation point is in a region where the second derivative is negative.

Computing c using a cubic approximation is better than bisection when both a and b are close to a minimum. However, the convergence can be slow when either a or b is far from a minimum since one of the bracketing points stays fixed while the other point creeps slowly toward the minimum at a linear rate. For example, consider the function $f(x) = x^2 - x^4$ which has a local minimum at $x = 0$ and consider the initial bracketing interval $[a_0, b_0] = [-0.1, 0.9]$. Computing c_k

Table 1. Numerical errors corresponding to (2) and (4) for the function $f(x) = x^2(1 - x)$

Values for a and b	Error for (2)	Error for (4)
$a = -0.5$, $b = 0.01, 0.02, \ldots, 2$	0.34×10^{-4}	0.061×10^{-4}
$a = -0.5$, $b = 0.01, 0.02, \ldots, 0.7$	0.17×10^{-5}	0.087×10^{-5}
$a = -0.5$, $b = 0.01, 0.02, \ldots, 0.25$	0.44×10^{-6}	0.40×10^{-6}
$b = -0.5$, $a = 0.01, 0.02, \ldots, 0.25$	0.86×10^{-6}	0.40×10^{-6}
$a = -0.5$, $b = 0.0025, 0.005, 0.0075, \ldots, 0.02$	0.52×10^{-6}	0.18×10^{-6}

Table 2. Computing c_k using (4) for the function $f(x) = x^2 - x^4$

k	a_k	b_k	c_k
0	−0.10000	0.90000	−0.04585
1	−0.04585	0.90000	−0.02073
2	−0.02073	0.90000	−0.00932
3	−0.00932	0.90000	−0.00417
4	−0.00417	0.90000	−0.00187
5	−0.00187	0.90000	−0.00083
6	−0.00083	0.90000	−0.00037
7	−0.00037	0.90000	−0.00016
8	−0.00016	0.90000	−0.00007
9	−0.00007	0.90000	−0.00003
10	−0.00003	0.90000	−0.00001

using (4) and updating the bracketing interval using (R_4), the right side of each interval stays fixed at $x = 0.9$ while the successive left sides creep toward zero.

In Table 2, we see that the distance between c_k and the true minimizer $x = 0$ is divided by about 2 in each iteration. The convergence is much slower than one hopes to achieve using a cubic approximation to f. This deficiency with cubic interpolation is reminiscent of a similar problem [11, p. 230] with *regula falsi*, a method for computing a root of an equation.

4. THE HYBRID METHOD

One method for improving the convergence speed is to insert a bisection step between each cubic interpolation step. However, a better strategy is to use the bracketing interval $[a_k, b_k]$ coupled with a_{k-1} to compute a better approximation c_k to a local minimizer of f. In each iteration of our algorithm, the interval $[a_k, b_k]$ satisfies (1) and the point a_k has the smallest function value of all the previously generated points. If the convergence is "slow" or the cost lacks convexity, a bisection step is performed. Conversely, if the convergence seems "fast" and the cost appears convex, then a cubic step is performed and the bracketing interval is updated using (R_3)–(R_5). In stating our algorithm, we utilize a parameter τ which is the specified error tolerance—the iterations stop when $|a_k - b_k| \leqslant \tau$. We also utilize the functions "cubic" and "step" and the subroutine "update" defined below:

Function cubic(a, b). This function returns the local minimizer (when it exists) for the Hermite cubic interpolant based on function values and derivatives at $x = a$ and at $x = b$. If the interpolating cubic is a linear function, then cubic$(a, b) = a$. If w is imaginary, in which case the cubic has no minimum, we replace w by zero in (4) to obtain a real approximation to a minimizer of the function being interpolated.

Function step(a, b, c, τ). This function forces the step size to be at least τ. Letting y denote minimum$[a, b]$ and letting z denote maximum$[a, b]$, return c if $y + \tau \leqslant c \leqslant z - \tau$. Otherwise, return $z - \tau$ if $c > (a + b)/2$ or return $y + \tau$ if $c \leqslant (a + b)/2$.

Subroutine update(a_k, b_k, c_k). This subroutine starts with a bracketing interval $[a_k, b_k]$ and a point c_k between a_k and b_k and generates a new bracketing interval $[a_{k+1}, b_{k+1}]$ using rules (R_3)–(R_5).

With these definitions, our univariate minimization scheme can be stated in the following way:

Cubic Algorithm

Step 1. If $|a_k - b_k| \leqslant \tau$, then stop. Otherwise, set $l_{k-1} = 2|a_k - b_k|$, set $\gamma = \text{cubic}(a_k, b_k)$, set $c_k = \text{step}(a_k, b_k, \gamma, \tau)$, and call update$(a_k, b_k, c_k)$.

Step 2. If $|a_{k+1} - b_{k+1}| \leqslant \tau$, then stop. Otherwise, set $l_k = l_{k-1}/2$. If $|c_k - a_k| > l_k$, then branch to step 5. If $|c_k - a_k| \leqslant l_k$, proceed to step 3.

Step 3. If $(f'(c_k) - f'(a_k))/(c_k - a_k) \leqslant 0$, then branch to step 5. Otherwise, proceed to step 4.

Step 4. Set $\gamma = \text{cubic}(c_k, a_k)$. If γ is outside the interval $[a_k, b_k]$, then branch to step 5. Otherwise, increment k by one, set $c_k = \text{step}(a_k, b_k, \gamma, \tau)$, call update$(a_k, b_k, c_k)$, and branch to step 2.

Step 5. Increment k by one, set $c_k = (a_k + b_k)/2$, call update(a_k, b_k, c_k), and branch to step 1 after incrementing k by one.

The technique used above to drive the width of the bracketing interval $[a_k, b_k]$ below the error tolerance τ is due to Brent [12]. Note that when the algorithm terminates after a cubic step, the final γ is often a better approximation to the desired minimizer than the point a_{k+1}. The reason for branching to the bisection step whenever $(f'(c_k)-f'(a_k))/(c_k-a_k) \leqslant 0$ is the following: Generally, the cubic step is most efficient near a local minimizer α for which $f''(\alpha) > 0$. The expressions we obtain for the error in the cubic approximation involve the second derivative of f. If this second derivative is zero or negative, then the point cubic(c_k, a_k) may even be farther from a local minimum than either c_k or a_k. If $(f'(c_k)-f'(a_k))/(c_k-a_k) \leqslant 0$, then the second derivative of f is nonpositive at a point between c_k and a_k. To ensure convergence, a bisection step is performed. Furthermore, when the second derivative of f vanishes at a local minimizer, or equivalently, at a multiple zero of f', the cubic iteration typically converges linearly. Below we show that the cubic iteration is closely related to Newton's method applied to the equation $f'(x) = 0$. It is well known that Newton's method converges linearly at a multiple zero and the linear convergence factor is $(m-1)/m$ when the multiplicity is m. Thus the convergence factor is $\geqslant 1/2$ when $m \geqslant 2$. By analogy, we expect that the convergence factor for the cubic iteration is $\geqslant 1/2$ when the second derivative of f vanishes at a local minimizer. Consequently, we expect that the bisection step will speed up the convergence of the Cubic Algorithm at a local minimizer where the second derivative of f vanishes.

5. CONVERGENCE

We now analyze the asymptotic convergence properties of the Cubic Algorithm when the error tolerance τ is zero. We begin with an alternative representation for some of the key terms in (4):

Lemma 3

If f is differentiable at $x = a$ and at $x = b$ and if p denotes the Hermite cubic interpolant of f based on the function value and derivative at $x = a$ and at $x = b$, then

$$\frac{f'(b)+v}{b-a} = \tfrac{1}{2}p''(b) \quad \text{and} \quad \frac{w^2}{(b-a)^2} = \tfrac{1}{4}p''(b)^2 - \tfrac{1}{2}p'(b)p'''(b), \tag{8}$$

where v and w are defined in (3).

Proof. Since p is the Hermite cubic interpolant of f, the fs embedded in the left side of each equality in (8) can be replaced by ps. Expanding the ps in a Taylor expansion about $x = b$ yields (8). □

Lemma 3 implies that cubic(a, b) approaches a limit as b approaches a when f is three times continuously differentiable near $x = a$ and the cubic that interpolates f and its first three derivatives at $x = a$ has a local a minimum. As a consequence, the function cubic(a, b) makes sense even when $b = a$; that is, cubic(a, a) is the limit of cubic(a, b) as b approaches a. Similarly, the convexity test "$(f'(c_k)-f'(a_k))/(c_k-a_k) \leqslant 0$" of step 3 should be replaced by the limiting relation "$f''(c_k) \leqslant 0$" whenever $c_k = a_k$. Note that when the error tolerance τ is positive, $c_k \neq a_k$ for every k and these limiting cases are never a concern. As an application of Lemma 3, we now show that the Cubic Algorithm converges to a point which satisfies the second order necessary conditions:

Theorem 1

Suppose that f is differentiable on an interval $[a_0, b_0]$ which satisfies (1) and $\tau = 0$. Then the a_k generated by the Cubic Algorithm approach a limit α contained in the interval $[a_0, b_0]$. If f is three times continuously differentiable in a neighborhood of α, then $f'(\alpha) = 0$ and $f''(\alpha) \geqslant 0$.

Proof. If the width of the bracketing intervals $[a_k, b_k]$ approaches zero as k increases, then by Lemma 1, the a_k approach a limit α which satisfies the second order necessary conditions. Conversely, suppose that the width of the bracketing intervals approaches a positive limit ω. Choose K sufficiently large that l_k is much smaller than ω for $k > K$ and no bisection steps are performed for $k > K$. In each iteration, either $a_{k+1} = c_k$ or $b_{k+1} = c_k$. If $b_{k+1} = c_k$, then the inequalities

$$\omega \leqslant |b_{k+1} - a_{k+1}| = |c_k - a_{k+1}| = |c_k - a_k| \leqslant l_k$$

yield a contradiction when $k > K$ since l_k is less than ω. Therefore, $a_{k+1} = c_k$ and $a_{k+1} = \text{cubic}(a_k, a_{k-1})$ for k sufficiently large. Since $l_{k+1} = l_k/2$ and since $|c_k - a_k| = |a_{k+1} - a_k| \leqslant l_k$ for each $k > K$, we conclude that the a_k approach a limit α. Utilizing one of the expressions in equation (4) (recall that each of these expressions is equivalent), rearranging terms in the identity $a_{k+1} = \text{cubic}(a_k, a_{k-1})$, and applying Lemma 3, we have

$$f'(a_k) = (a_{k+1} - a_k)\{\tfrac{1}{2}p''(a_k) + [\tfrac{1}{4}p''(a_k)^2 - \tfrac{1}{2}p'(a_k)p'''(a_k)]^{1/2}\}, \tag{9}$$

where p is the Hermite cubic interpolant based on function values and derivatives at $x = a_k$ and $x = a_{k-1}$. By repeated applications of Rolle's theorem (see [13, p. 190]), the third derivative of p is equal to the third derivative of f at some point between a_k and a_{k-1}. As a consequence, each of the derivatives contained in (9) approach the corresponding derivatives of f at α as k increases. Since $|a_{k+1} - a_k|$ tends to zero as k increases, (9) implies that $f'(\alpha) = 0$. Since the convexity test is passed for $k > K$, the ratio $(f'(a_{k+1}) - f'(a_k))/(a_{k+1} - a_k)$ is positive for k sufficiently large. Since this ratio is equal to the second derivative of f at some point between a_{k+1} and a_k, it follows that $f''(\alpha) \geqslant 0$. □

To show that the Cubic Algorithm is quadratically convergent, we must analyze the error in the cubic iteration $a_{k+1} = \text{cubic}(a_k, a_{k-1})$. By Lemma 3 and formula (4), the iteration $a_{k+1} = \text{cubic}(a_k, a_{k-1})$ is closely related to the Newton iteration

$$n_{k+1} = n_k - f'(n_k)/f''(n_k).$$

In addition, as we now show, the convergence properties of the cubic iteration are similar to the convergence properties of Newton's method.

Corollary 1

If f is four times continuously differentiable in a neighborhood of a local minimizer α of f and $f''(\alpha)$ is positive, then for a_0 and a_1 sufficiently close to α, the iteration $a_{k+1} = \text{cubic}(a_k, a_{k-1})$ converges to α and there exists a constant r, independent of k, such that

$$|e_{k+1}| \leqslant r|e_k|e_{k-1}^2, \tag{10}$$

where $e_k = a_k - \alpha$. Moreover, we have

$$\lim_{k\to\infty} \frac{|a_{k+1} - a_k|}{|a_k - a_{k-1}|} = 0. \tag{11}$$

Proof. Subtracting α from each side of the equation $a_{k+1} = \text{cubic}(a_k, a_{k-1})$, utilizing one of the expressions in (4), applying Lemma 3, and applying the mean value theorem to $f'(a_k) = f'(a_k) - f'(\alpha)$, we have

$$e_{k+1} = \left(1 - \frac{f''(\theta)}{\tfrac{1}{2}p''(a_k) + \sqrt{\tfrac{1}{4}p''(a_k)^2 - \tfrac{1}{2}p'(a_k)p'''(a_k)}}\right)e_k, \tag{12}$$

where p denotes the cubic which interpolates f and its derivative at $x = a_k$ and $x = a_{k-1}$ and θ lies between α and a_k. Since each derivative of p up to the third derivative is equal to a corresponding derivative of f at some point between a_k and a_{k-1} (this follows from repeated applications of Rolle's theorem as in [13, p. 190]) and since $f'(\alpha) = 0$, (12) implies that $|e_{k+1}| \leqslant 0.5|e_k|$ whenever a_k and a_{k-1} are sufficiently close to α. Hence, the iterations contract in a neighborhood of α and there is no loss of generality in assuming that $|e_{k+1}| \leqslant 0.5|e_k|$ and $|e_k| \leqslant 0.5|e_{k-1}|$. The standard estimate for the error in the derivative associated with Hermite cubic interpolation (see [13, p. 190]) yields

$$f'(a_{k+1}) = f'(a_{k+1}) - p'(a_{k+1}) = \tfrac{1}{6}(a_{k+1} - a_k)(a_{k+1} - a_{k-1})(a_{k+1} - \xi)f^{(4)}(\eta),$$

where ξ lies between a_k and a_{k-1} and η lies in the convex hull of the points a_{k+1}, a_k, and a_{k-1}. Again, by the mean value theorem, $f'(a_{k+1}) = f''(\bar{\theta})e_{k+1}$ where $\bar{\theta}$ lies between a_{k+1} and α. Hence, we have

$$f''(\bar{\theta})e_{k+1} = \tfrac{1}{6}(a_{k+1} - a_k)(a_{k+1} - a_{k-1})(a_{k+1} - \xi)f^{(4)}(\eta). \tag{13}$$

By the triangle inequality and the fact that the iterations contract, $|a_{k+1}-a_k| \leqslant |e_{k+1}| + |e_k| \leqslant 1.5|e_k|$, $|a_{k+1}-a_{k-1}| \leqslant 1.25|e_{k-1}|$, and $|a_{k+1}-\xi| \leqslant 1.25|e_{k-1}|$. These inequalities, the relation $f''(\alpha) > 0$, and (13) give (10). Also, the triangle inequality, (10), and the contraction property $|e_k| \leqslant 0.5|e_{k-1}|$ imply that

$$\frac{|a_{k+1}-a_k|}{|a_k-a_{k-1}|} \leqslant \frac{|e_{k+1}|+|e_k|}{|e_{k-1}|-|e_k|} \leqslant \frac{r|e_k|e_{k-1}^2 + r|e_{k-1}|e_{k-2}^2}{0.5|e_{k-1}|} = 2r(|e_k e_{k-1}| + e_{k-2}^2).$$

Since the right side tends to zero, (11) holds. □

In related work [14], Tamir establishes (10) but he assumes that f is five times continuously differentiable. The analysis above shows how to weaken this continuity assumption. Let us now consider the convergence rate of the cubic iteration.

Corollary 2

If f is four times continuously differentiable in a neighborhood of a local minimizer α of f and $f''(\alpha)$ is positive, then for a_0 and a_1 sufficiently close to α, the iteration $a_{k+1} = \text{cubic}(a_k, a_{k-1})$ converges quadratically to α. That is, there exist constants p and q, independent of k, such that $|a_k - \alpha| \leqslant q/p^{2^k}$ for every k where $p > 1$.

Proof. Let us consider a neighborhood of α where (10) holds and let q denote any positive number with the property that $rq^2 < 1$. If

$$\max\{|a_0-\alpha|, |a_1-\alpha|\} \leqslant q/4,$$

then the error $e_k = a_k - \alpha$ satisfies the relation $|e_k| \leqslant q/2^{2^k}$ for $k = 0$ and for $k = 1$. Proceeding by induction, suppose that $|e_k| \leqslant q/2^{2^k}$ and $|e_{k-1}| \leqslant q/2^{2^{k-1}}$. By (10) and the condition $rq^2 < 1$, we conclude that $|e_{k+1}| \leqslant q/2^{2^{k+1}}$ so the proof is complete. □

Using the terminology of Ortega and Rheinboldt (see [15, p. 290]), Corollary 2 implies that the root convergence order of iteration $a_{k+1} = \text{cubic}(a_k, a_{k-1})$ is at least 2. In related work [14]. Tamir shows that the root convergence order of the cubic iteration is at least $\sqrt{3}$, which is weaker than the conclusion of Corollary 2. He also shows that the quotient convergence order of the cubic iteration is two if f has five continuous derivatives near α and $f^{(4)}(\alpha) \neq 0$. This conclusion is slightly stronger than the conclusion of Corollary 2, but the assumptions in [14] are also stronger. We do not require that $f^{(4)}(\alpha) \neq 0$ and we just need four continuous derivatives. Moreover, if $f^{(4)}(\alpha) = 0$, and f has five continuous derivatives near α, then from the proof of Corollary 1, we see that the inequality (10) can be strengthened to $|e_{k+1}| \leqslant r|e_k e_{k-1}^3|$. Consequently, the error estimate in Corollary 2 can be strengthened to $|e_k| \leqslant q/p^{\sigma^k}$ where $\sigma = (1+\sqrt{13})/2 \approx 2.3$. In other words, when $f^{(4)}(\alpha) = 0$, the root convergence order of the cubic iteration is at least $(1+\sqrt{13})/2$.

In analyzing the convergence rate of the Cubic Algorithm, we also need the following result connecting function values and the error:

Lemma 4

If f is two times continuously differentiable in a neighborhood of a local minimizer α of f and $f''(\alpha)$ is positive, then there exists a neighborhood $\mathbf{N}$ of α with the property that for each w, x, y, and z in $\mathbf{N}$ with $f(w) \leqslant f(x)$ and $2|y-\alpha| \leqslant |z-\alpha|$, we have $|w-\alpha| \leqslant 2|x-\alpha|$ and $f(y) \leqslant f(z)$. These inequalities are strict except for the special cases $w = x = \alpha$ or $y = z = \alpha$.

Proof. Let I denote a closed interval containing α in its interior, let m denote the minimum of f'' on I, and let M denote the maximum of f'' on I. We assume that I is small enough that m is positive. Expanding f in a Taylor series about α, we have

$$f(x) \geqslant f(\alpha) + \frac{m}{2}(x-\alpha)^2 \quad \text{and} \quad f(y) \leqslant f(\alpha) + \frac{M}{2}(y-\alpha)^2. \tag{14}$$

If $f(x) \leqslant f(y)$, it follows that

$$|x-\alpha| \leqslant \left(\frac{M}{m}\right)^{1/2} |y-\alpha|.$$

Table 3. Iterations for the Cubic Algorithm and the function $f(x) = x^2 - x^4$

k	a_k	b_k	c_k
0	−0.1000000000000	0.9000000000000	−0.0458581335842
1	−0.0458581335842	0.9000000000000	−0.0006492938846
2	−0.0006492938846	0.9000000000000	−0.0000013817061
3	−0.0000013817061	0.9000000000000	−0.0000000000005

Choosing I so small that $M/m < 4$, we conclude that $|x-\alpha| \leqslant 2|y-\alpha|$. On the other hand, if $2|y-\alpha| \leqslant |x-\alpha|$, then the sum of the inequalities in (14) gives us the relation

$$f(y)-f(x) \leqslant \frac{M}{2}(y-\alpha)^2 - \frac{m}{2}(x-\alpha)^2 \leqslant \left(\frac{M}{8}-\frac{m}{2}\right)(x-\alpha)^2.$$

Again, if I is so small that $M/m < 4$, it follows that $M/8 - m/2 < 0$ and $f(y) \leqslant f(x)$. □

Finally, we give our quadratic convergence results for the cubic algorithm.

Theorem 2

Suppose that the a_k generated by the Cubic Algorithm converge to a local minimizer α of f, f is four times continuously differentiable in a neighborhood of α, and $f''(\alpha)$ is positive. Then the a_k converge quadratically to α.

Proof. First, let us assume that no bisections are performed for k sufficiently large. Let **N** denote a neighborhood of α for which the hypotheses of Lemma 4 are satisfied and for which the following additional properties hold: The second derivative of f is positive throughout **N** and if a and b lie in **N** and $c = \text{cubic}(a, b)$, then $|c-\alpha| \leqslant 0.25|a-\alpha|$ (the existence of this neighborhood was established during the proof of Corollary 1). Choose K sufficiently large that a_k lies in **N** and no bisections are performed for $k \geqslant K$. If $a_{k+1} = c_k$ for some $k > K$, then defining $\gamma = \text{cubic}(c_k, a_k)$, we have $|\gamma-\alpha| \leqslant 0.25|c_k-\alpha| = 0.25|a_{k+1}-\alpha|$. Since $f(a_{k+1}) \leqslant f(b_{k+1})$, Lemma 4 tells us that $0.5|a_{k+1}-\alpha| \leqslant |b_{k+1}-\alpha|$. Hence, γ lies between a_{k+1} and b_{k+1} and $c_{k+1} = \gamma$. Also, by Lemma 4, $f(\gamma) < f(a_{k+1})$ (except when $a_{k+1} = \alpha$ and the iterations terminate) and by rule (R_4), $a_{k+2} = c_{k+1}$. Proceeding by induction, $a_{k+2} = c_{k+1} = \text{cubic}(c_k, a_k) = \text{cubic}(a_{k+1}, a_k)$ for each successive k. Since $a_{k+2} = \text{cubic}(a_{k+1}, a_k)$, it follows from Corollary 2 that the convergence is quadratic. On the other hand, suppose that $b_{k+1} = c_k$ for every $k > K$. By rules (R_3)–(R_5), $a_{k+1} = a_k$ for every $k > K$. Since the a_k converge to α, $a_k = \alpha$ for every $k > K$. Since $f'(\alpha) = 0$, $\text{cubic}(c_k, a_k) = \alpha$ for k sufficiently large and the algorithm converges in a finite number of iterations.

Now consider the case where an infinite number of bisections are performed. Suppose that an initialization step is performed at iteration K and both a_k and b_k lie in **N**. By the argument earlier, $a_{K+1} = c_K = \text{cubic}(a_K, b_K)$ and $a_{K+2} = c_{K+1} = \text{cubic}(a_{K+1}, a_K)$. By (11) with a_{k-1} identified with b_k, we conclude that the next iteration will not branch to the bisection step due to slow convergence if K is sufficiently large. And since f'' is positive in **N**, the convexity test is passed for each $k > K$. Proceeding by induction, $a_{k+2} = \text{cubic}(a_{k+1}, a_k)$ for k sufficiently large, and by Corollary 2, the convergence is quadratic. □

To illustrate the convergence that is typically observed with the Cubic Algorithm, Table 3 gives iterations for the function $f(x) = x^2 - x^4$ and the initial bracketing interval $[a_0, b_0] = [-0.1, 0.9]$.

6. THE CONJUGATE GRADIENT METHOD

In this section, we discuss how to incorporate the Cubic Algorithm in the conjugate gradient method. Let f denote a real valued function of n variables. The Fletcher–Reeves formulation of the conjugate gradient method can be expressed

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k \quad \text{and} \quad \mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$$

where $\mathbf{g}_k = \nabla f(\mathbf{x}_k)^{\mathrm{T}}$, the initial search direction $\mathbf{d}_0$ is $-\mathbf{g}_0$,

$$s_k = \arg\min_{s \geqslant 0} f(\mathbf{x}_k + s\mathbf{d}_k) \quad \text{and} \quad \beta_k = \frac{\mathbf{g}_{k+1}^{\mathrm{T}} \mathbf{g}_{k+1}}{\mathbf{g}_k^{\mathrm{T}} \mathbf{g}_k}.$$

Notice that each conjugate gradient iteration involves a one dimensional minimization. In an efficient implementation of the conjugate gradient method, this minimization is performed with the lowest accuracy that preserves the convergence properties of the conjugate gradient method. We will consider three issues: global convergence, quadratic convergence, and descent.

Various steplength rules have been devised to guarantee the convergence of a multivariate scheme. Some examples (see [8]) are Armijo's rule, Goldstein's test, and Wolfe's test. In our implementation of the conjugate gradient method, we start by finding a point that satisfies either Armijo's or Goldstein's criterion. Let $\phi(s)$ denote the function $f(\mathbf{x}_k + s\mathbf{d}_k)$. Given parameters $\mu \in (0, 1)$ and $\rho > 1$, the step $s = a$ satisfies Armijo's criterion if

$$\phi(a) \leqslant \phi(0) + \mu\phi'(0)a \quad \text{and} \quad \phi(\rho a) \geqslant \phi(0) + \mu\phi'(0)\rho a. \tag{15}$$

Given a parameter $\lambda \in (0, 0.5)$, the step $s = a$ satisfies Goldstein's criterion if

$$\phi(a) \leqslant \phi(0) + \lambda\phi'(0)a \quad \text{and} \quad \phi(a) \geqslant \phi(0) + (1 - \lambda)\phi'(0)a. \tag{16}$$

A point that satisfies Armijo's criterion is generated by successively multiplying or dividing a starting a by the factor ρ until (15) holds. As long as ϕ is bounded from below and $\phi'(0) < 0$, a point will be generated that satisfies (15). A disadvantage with Armijo's criterion is that at least two function evaluations are required to determine whether (15) holds. In contrast, Goldstein's test may be satisfied by the starting a in which case one function evaluation is needed. To implement this combined Armijo–Goldstein strategy, we take (for convenience) $\mu = \lambda < 0.5$ and we test whether (16) holds for a given starting a. If (16) holds, then we stop. Otherwise, there are two separate cases to consider;

(a) If $\phi(a) \leqslant \phi(0) + \lambda\phi'(0)a$, then a is successively multiplied by ρ until an a is generated that satisfies either (15) or (16). In particular, evaluating $\phi(a_k)$ for $k = 0, 1, \ldots$ where $a_k = \rho^k a$, we stop at the first k for which $\phi(a_k) \geqslant \phi(0) + (1 - \lambda)\phi'(0)a_k$. If $\phi(a_k) \leqslant \phi(0) + \lambda\phi'(0)a_k$, then $a = a_k$ satisfies Goldstein's criterion. Conversely, if $\phi(a_k) > \phi(0) + \lambda\phi'(0)a_k$, then $a = a_{k-1}$ satisfies Armijo's criterion.

(b) If $\phi(a) > \phi(0) + \lambda\phi'(0)a$, then a is successively divided by ρ until an a is generated that satisfies (15).

Now let us consider quadratic convergence. There is an enormous literature dealing with quadratic convergence of the conjugate gradient method. Some of the pioneering work appears in papers by Cohen [16] and by Daniel [17–19]. More recent papers include [20] by Klessig and Polak, [21, 22] by Lenard, and [23, 24] by Baptist and Stoer. In Cohen's paper [16], he demonstrates that the conjugate gradient algorithm is quadratically convergent in the sense that n iterations essentially square the error. He proves this quadratic convergence result for the Daniel, Fletcher–Reeves, and Polak-Ribière formulations of the conjugate gradient method with exact line search (see [8] for a statement of various formulations of the conjugate gradient method). More recent conjugate gradient papers treat inaccurate line searches. These papers typically demonstrate that conjugate gradient methods (or more general classes of methods) are quadratically convergent if the line search is performed with enough accuracy so that the function derivative satisfies some bound. Thus to implement the conjugate gradient method, we can apply the Cubic Algorithm until the derivative satisfies one of these bounds.

As an alternative to this brute force application of the Cubic Algorithm, we consider a special quadratic interpolation step. In Daniel's explicit conjugate gradient scheme, the step s_k is given by

$$s_k = \frac{-\mathbf{g}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{H}_k \mathbf{d}_k},$$

where $\mathbf{H}_k$ is the Hessian of f evaluated at $\mathbf{x}_k$. Notice that this s_k corresponds to one step of Newton's method applied to the equation $\phi'(s) = 0$ starting from $s = 0$. Since one step of Newton's method essentially squares the error near a local minimizer and since the Daniel step apparently preserves the overall quadratic convergence rate of the conjugate gradient scheme, this suggests that in the Fletcher–Reeves formulation, one step of any quadratically convergent scheme will yield sufficient accuracy in the line search to preserve the quadratic convergence of the conjugate gradient method.

In addition, numerical experiments as well as a recent convergence analysis by a Ph.D. student Holly Hirst all indicate that the quadratic convergence rate of the conjugate gradient algorithm is preserved if one step of any quadratically convergent algorithm is used in the line search. Although the standard quadratic interpolation algorithm for univariate minimization has convergence order around 1.3, the following lemma implies that for an approximation derived from a special quadratic interpolation, the error is squared.

Lemma 5

Given a point b, suppose that ϕ is three times continuously differentiable on the interval $[b, \infty)$ and let $m(s)$ and $M(s)$ denote the minimum and the maximum of the second derivative on the interval $[b, s]$. If $\phi'(b) \leq 0$ and $m(a) > 0$ for some $a > b$, then the quadratic that interpolates the value and the derivative of ϕ at $x = b$ and the value of ϕ at $x = a$ has a minimizer $q \leq b - \phi'(b)/m(a)$. If in addition ϕ has a local minimizer $\alpha > b$, $m(\alpha) \geq 0$, and $M(\alpha)/m(a) \leq 2$, then $|q - \alpha| \leq |b - \alpha|$. If in addition $M(\max\{a, \alpha\})/m(\max\{a, \alpha\}) \leq 4$ and $\phi(a) \leq \phi(b)$, then

$$|q - \alpha| \leq \frac{3(b - \alpha)^2}{m(\max\{\alpha, q\})} \max_{b \leq s \leq \max\{a, q\}} |\phi'''(s)|.$$

The key condition in Lemma 5 that ensures the error is squared is the inequality $\phi(a) \leq \phi(b)$.

Proof. Let p denote the quadratic interpolant of ϕ based on the function value and derivative at $x = b$ and the function value at $x = a$. If $p'' > 0$, then this quadratic has a minimizer q given by $q = b - p'(b)/p''$. Since the second derivative of this interpolant is equal to the second derivative of ϕ at some point σ between a and b, it follows that

$$q = b - \phi'(b)/\phi''(\sigma), \tag{17}$$

which implies that $q \leq b - \phi'(b)/m(a)$ when $\phi'(b) \leq 0$. Subtracting α from each side of (17) and expanding the first derivative in a Taylor series about α, we obtain

$$q - \alpha = \left[1 - \frac{\phi''(\theta)}{\phi''(\sigma)}\right](b - \alpha), \tag{18}$$

where θ lies between b and α. If the bracketed expression in (18) is at most one in magnitude, then $|q - \alpha| \leq |b - \alpha|$. Observe that the relations $M(\alpha)/m(a) \leq 2$, $m(\alpha) \geq 0$, and $m(a) > 0$ ensure that the bracketed expression in (18) has magnitude at most one. As in the proof of Corollary 1, the fact that q minimizes a quadratic interpolant of ϕ implies that

$$\phi''(\bar{\theta})(q - \alpha) = \tfrac{1}{2}(q - b)(q - \xi)\phi'''(\eta) \tag{19}$$

where $\bar{\theta}$ lies between q and α, ξ lies between a and b, and η lies in the convex hull of a, b, and q. If $|q - \alpha| \leq |b - \alpha|$, then $|q - b| \leq |q - \alpha| + |b - \alpha| \leq 2|b - \alpha|$. If $M(\max\{a, \alpha\})/m(\max\{a, \alpha\}) \leq 4$, then by Lemma 4 and the assumption that $\phi(a) \leq \phi(b)$, we conclude that $|a - \alpha| \leq 2|b - \alpha|$ and $|q - a| \leq |q - \alpha| + |a - \alpha| \leq 3|b - \alpha|$. Since ξ lies between a and b, it again follows from the triangle inequality that $|q - \xi| \leq 3|b - \alpha|$. These inequalities in conjunction with (19) complete the proof. □

In formulating a globally convergent conjugate gradient algorithm, we utilize the Armijo–Goldstein strategy (a) and (b) above to obtain a point a such that $\phi(a) \leq \phi(0)$. Forming the quadratic that interpolates the function value and derivative at $s = 0$ and the function value at $s = a$, we compute the minimizer q if it exists:

$$q = \frac{-\phi'(0)a}{2\left[\dfrac{\phi(a) - \phi(0)}{a} - \phi'(0)\right]}. \tag{20}$$

By Lemma 5, we know that under appropriate assumptions, the error $|q - \alpha||\mathbf{d}_k|$ associated with the step $s = q$ is on the order of the square of the error $|\alpha||\mathbf{d}_k|$ corresponding to $s = 0$. If $\phi(q) \leq \phi(a)$, then we take $s_k = q$ as our approximation to a local minimizer along the search direction. If $\phi(q) > \phi(a)$, then $s_k = a$ is the approximation.

There is still one more issue to contend with when implementing the conjugate gradient method. Unless the local minimizer along each search direction is computed with sufficient accuracy, the new direction $\mathbf{d}_{k+1}$ may not be a decent direction. When an exact local minimizer is computed along a search direction, the relation $\phi'(s_k) = \mathbf{g}_{k+1}^T \mathbf{d}_k = 0$ holds and the formula $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$ implies that $\mathbf{g}_{k+1}^T \mathbf{d}_{k+1} = -\mathbf{g}_{k+1}^T \mathbf{g}_{k+1} \leqslant 0$. Thus the new direction is a descent direction when $\mathbf{g}_{k+1} \neq 0$. On the other hand, for an approximate local minimizer, $\mathbf{g}_{k+1}^T \mathbf{d}_k \neq 0$ and $\mathbf{g}_{k+1}^T \mathbf{d}_{k+1} \neq -\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}$. To guarantee that the new direction is a descent direction, we require that $\mathbf{g}_{k+1}^T \mathbf{d}_{k+1} \leqslant -\epsilon \mathbf{g}_{k+1}^T \mathbf{g}_{k+1}$ where $0 < \epsilon < 1$. Referring to the definition of $\mathbf{d}_{k+1}$, this condition is equivalent to

$$\phi'(s_k) = \mathbf{d}_k^T \mathbf{g}_{k+1} \leqslant (1-\epsilon)|\mathbf{g}_k|^2. \tag{21}$$

If the quadratic approximation generated by (20) satisfies (21), then we proceed to the next conjugate gradient iteration. Conversely, if (21) is violated, then the Cubic Algorithm can be used to generate a point that satisfies (21). In particular, since $\phi'(s_k) > (1-\epsilon)|\mathbf{g}_k|^2 \geqslant 0$ when (21) is violated, an initial bracketing interval for the Cubic Algorithm can be established on the interval $[0, s_k]$ (be sure to take advantage of all the previous function values to make this interval as small as possible).

In summary, for the Fletcher–Reeves formulation of the conjugate gradient method, the univariate search along the direction $\mathbf{d}_k$ can be implemented in the following way:

Conjugate gradient search scheme

Step 1. Using either procedure (a) or (b), find a point that satisfies either (15) or (16).

Step 2. Using formula (20) and the point a with smallest function value generated in step 1, obtained an approximation q to a local minimizer of ϕ. If $\phi(q) \leqslant \phi(a)$, then $s_k = q$ is our approximation to a local minimizer along the search direction. If either $\phi(q) > \phi(a)$ or q does not exist, then $s_k = a$ is our approximation.

Step 3. If the approximation s_k (generated in step 2) to the local minimizer satisfies (21), then proceed to the next conjugate gradient iteration. Otherwise, perform iterations of the Cubic Algorithm until (21) holds.

One final suggestion: In step 1, we have not explained how to generate the starting guess employed in either (a) or (b). Typically, the step length associated with one conjugate gradient iteration becomes the starting guess for the next iteration. In some cases, however, this starting guess can be in error by many orders of magnitude. In practice, we have found that some function evaluations in step 1 can be eliminated if the starting guess is the minimizer of a quadratic fit based on the function value and derivative at $s = 0$ and the function value at $s = \theta s_{k-1}$ where s_{k-1} is the step employed in the previous iteration and $0 < \theta < 1$. To summarize, the following step can be inserted before step 1:

Step 0. If s_{k-1} is the step employed in the previous iteration, then the q given by (20) with $a = \theta s_{k-1}$ is the starting guess for step 1 if q exists. (If $\phi(\theta s_{k-1}) \leqslant \phi(0)$, then after completing step 1, step 2 can be skipped.)

We now prove a global convergence result for the Conjugate Gradient Search Scheme. This theorem demonstrates that a convergent subsequence approaches a point where the gradient vanishes. After the proof, we observe that with small changes in the assumptions, the entire sequence approaches a local minimizer. Note that Al-Baali [25] gives a global convergence result for the Fletcher–Reeves formulation of the conjugate gradient method and for a line search that satisfies a Goldstein–Wolfe–Powell condition. His analysis does not seem to apply to the Conjugate Gradient Search Scheme due to differences in the termination conditions. For example, in [1] the analogue of (21) involves the absolute value of $\phi'(s_k)$. Also, in the Conjugate Gradient Search Scheme, the final s_k may not satisfy the first condition in either (15) or (16).

Theorem 3

Let $\mathbf{x}_0$ denote the starting guess of the conjugate gradient method and suppose that f is two times continuously differentiable on the convex hull $\mathbf{F}$ of the level set $\{\mathbf{x}: f(\mathbf{x}) \leqslant f(\mathbf{x}_0)\}$, the spectral radius

of the Hessian of f is uniformly bounded on $\mathbf{F}$, f is bounded from below on $\mathbf{F}$, and the search directions in the Conjugate Gradient Search Scheme are renormalized to be the negative gradient every r iterations for some r. Then any convergent subsequence of the $\mathbf{x}_{kr}$ converges to a point where the gradient of f vanishes.

Proof. Let a_k denote the point generated in step 1 (corresponding to iteration k of the conjugate gradient method) which satisfies either (15) or (16) and let M denote a bound for the spectral radius of the Hessian of f over $\mathbf{F}$. If a_k satisfies the second inequality in (15), then expanding $\phi(\rho a_k)$ in a Taylor series about $s = 0$ yields

$$\phi(0) + \mu\phi'(0)\rho a_k \leqslant \phi(\rho a_k) \leqslant \phi(0) + \phi'(0)\rho a_k + \frac{M}{2}\rho^2 a_k^2 |\mathbf{d}_k|^2.$$

Since $\phi'(0) = \mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k \leqslant -\epsilon|\mathbf{g}_k|^2$, it follows that

$$a_k \geqslant \frac{2(1-\mu)\epsilon}{M\rho}\frac{|\mathbf{g}_k|^2}{|\mathbf{d}_k|^2}.$$

In a similar fashion, if the second inequality in (16) holds, then

$$a_k \geqslant \frac{2\lambda\epsilon}{M}\frac{|\mathbf{g}_k|^2}{|\mathbf{d}_k|^2}.$$

Combining these lower bounds for a_k, we have $a_k \geqslant m|\mathbf{g}_k|^2/|\mathbf{d}_k|^2$ where m is the minimum of the expressions $2(1-\mu)\epsilon/M\rho$ and $2\lambda\epsilon/M$. Since step 2 and step 3 maintain descent, the final step s_k associated with iteration k of the Conjugate Gradient Search Scheme has the property that $f(\mathbf{x}_{k+1}) = \phi(s_k) \leqslant \phi(a_k)$. Combining the first inequality in (15) or (16) and the inequality $\phi'(0) \leqslant -\epsilon|\mathbf{g}_k|^2$ with the lower bound for a_k gives $f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{x}_k) - \beta|\mathbf{g}_k|^4/|\mathbf{d}_k|^2$ where $\beta = \min\{\epsilon m\lambda, \epsilon m\mu\}$. Summing these inequalities over k yields:

$$f(\mathbf{x}_{kr+1}) \leqslant f(\mathbf{x}_0) - \beta\sum_{i=0}^{kr}\frac{|\mathbf{g}_i|^4}{|\mathbf{d}_i|^2} \leqslant f(\mathbf{x}_0) - \beta\sum_{i=0}^{k}\frac{|\mathbf{g}_{ir}|^4}{|\mathbf{d}_{ir}|^2} = f(\mathbf{x}_0) - \beta\sum_{i=0}^{k}|\mathbf{g}_{ir}|^2, \tag{22}$$

where $\mathbf{d}_{ir} = -\mathbf{g}_{ir}$. Since f is bounded from below on $\mathbf{F}$, we conclude that the gradients $\mathbf{g}_{kr}$ tend to zero as k increases. □

Corollary 3

Suppose that the iterations generated by the Conjugate Gradient Search Scheme are contained in a compact convex set $\mathbf{K}$ where f is two times continuously differentiable, the search directions are renormalized to be the negative gradient every r iterations for some r, and there exists an upper bound M and a lower bound $m > 0$ for the eigenvalues of the Hessian of f on $\mathbf{K}$. Then the iterations generated by the Conjugate Gradient Search Scheme approach the unique minimizer for f on the set $\mathbf{K}$.

Proof. By Theorem 3 and the compactness of $\mathbf{K}$, a subsequence of the iterations approach a point $\mathbf{x}^*$ where the gradient vanishes. Since f is convex on $\mathbf{K}$, $\mathbf{x}^*$ is the unique global minimizer of f on $\mathbf{K}$. By the mean value theorem, we have $|\mathbf{g}_{k+1} - \mathbf{g}_k| \leqslant M|\mathbf{x}_{k+1} - \mathbf{x}_k|$, which implies that

$$|\mathbf{g}_{k+1}| \leqslant |\mathbf{g}_k| + M|\mathbf{x}_{k+1} - \mathbf{x}_k| = |\mathbf{g}_k| + Ms_k|\mathbf{d}_k|. \tag{23}$$

Expanding f in a Taylor series about $\mathbf{x}_k$ yields

$$f(\mathbf{x}_{k+1}) \geqslant f(\mathbf{x}_k) + \mathbf{g}_k^{\mathrm{T}}(\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{m}{2}|\mathbf{x}_{k+1} - \mathbf{x}_k|^2 = f(\mathbf{x}_k) + s_k\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k + \frac{m}{2}s_k^2|\mathbf{d}_k|^2.$$

Since $f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{x}_k)$, it follows that

$$ms_k^2|\mathbf{d}_k|^2 \leqslant -2s_k\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k \leqslant 2s_k|\mathbf{g}_k|\,|\mathbf{d}_k|,$$

which implies that $s_k|\mathbf{d}_k| \leqslant 2|\mathbf{g}_k|/m$. Referring to (23), we conclude that

$$|\mathbf{g}_{k+1}| \leqslant \left[1 + \frac{2M}{m}\right]|\mathbf{g}_k|. \tag{24}$$

Since $\mathbf{g}_{i_r}$ tends to zero, (24) shows that all the $\mathbf{g}_k$ approach zero. Finally, expanding $f(\mathbf{x}^*)$ in a Taylor series about $\mathbf{x}_k$ gives

$$f(\mathbf{x}^*) \geq f(\mathbf{x}_k) + \mathbf{g}_k^T(\mathbf{x}^* - \mathbf{x}_k) + \frac{m}{2}|\mathbf{x}_k - \mathbf{x}^*|^2.$$

Since $f(\mathbf{x}^*) \leq f(\mathbf{x}_k)$, it follows that

$$\frac{m}{2}|\mathbf{x}_k - \mathbf{x}^*|^2 \leq \mathbf{g}_k^T(\mathbf{x}_k - \mathbf{x}^*) \leq |\mathbf{g}_k||\mathbf{x}_k - \mathbf{x}^*|,$$

and $|\mathbf{x}_k - \mathbf{x}^*| \leq 2|\mathbf{g}_k|/m$. Since the $\mathbf{g}_k$ tend to zero, the $\mathbf{x}_k$ approach $\mathbf{x}^*$. □

7. NUMERICAL COMPARISONS

To compare the Conjugate Gradient Search Scheme to various other algorithms, we have solved some of the standard test problems. In [26] Ecker and Kupferschmid compare the performance of various constrained optimization algorithms. The following algorithms were studied: EA3: the variant of the ellipsoid algorithm implemented by Kupferschmid and Ecker [27]. GRG2: The generalized reduced gradient algorithm of Lasdon *et al.* [28]. IQP: The Han [29]–Powell [30] iterative quadratic programming algorithm implemented by Crane *et al.* [31]. NAG8: The augmented Lagrangian algorithm of Gill and Murray [32] as implemented in subroutine EO4VAF of the Mark 8 NAG Subroutine Library. RQP: The recursive (iterative) quadratic programming method of Biggs [33] as implemented in subroutine OPRQP of the Hatfield Subroutine Library [34].

Although most of the test problems studied in [26] contained constraints, the unconstrained problem Colville 4 [35] was examined. In Colville 4 the cost function is

$$f(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1).$$

This problem, proposed by C. F. Wood at Westinghouse Research Laboratory, is essentially two Rosenbrock functions tied together and the minimizer is $x_1 = x_2 = x_3 = x_4 = 1$. For an enlightening picture of the Rosenbrock function see [36, p. 5]. Using the starting condition $x_1 = x_2 = x_3 = x_4 = 0$ and the stopping criterion $|f(\mathbf{x}_k) - f(\mathbf{x}^*)|/|f(\mathbf{x}_0) - f(\mathbf{x}^*)| \leq 0.001$ where $\mathbf{x}^*$ denotes the solution and $\mathbf{x}_0$ is the starting guess, Ecker and Kupferschmid obtained the results given in Table 4.

For comparison, the Conjugate Gradient Search Scheme, with $\lambda = \mu = \varepsilon = 0.1$, $\rho = 5$, and $\theta = 0.3$, satisfied the convergence criterion after 7 iterations, 20 function evaluations, and 9 gradient evaluations. Since there are four unknowns in Colville 4, the direction vector was renormalized to be the negative gradient after every four consecutive iterations. A copy of the program used in these experiments is stored under the name CG in the subroutine package NAPACK. Instructions for obtaining a copy of this program by electronic mail appear in [37].

It should be noted that although our program seems to perform spectacularly relative to the data in Table 4, one must be cautious in drawing conclusions. First, the algorithms in Table 4 are general constrained optimization routines while the Conjugate Gradient Search Scheme is an unconstrained optimization routine. Second, Colville 4 is a somewhat unstable test problem in the following sense: If the iterations turn down the wrong valley (in four dimensions) looking for the minimizer, it often takes many iterations before the iterations turn around and head up the correct valley. Moreover, tiny changes in the starting point can cause the iterations to turn down the wrong valley.

Some experimental results that focus on conjugate gradient schemes appear in [38]. In these experiments, 10 different conjugate gradient codes are examined: Harwell subroutines VA14A and

Table 4. Algorithm statistics extracted from [26] for Colville 4

Method	Iterations	Time (s)	Function evaluations	Gradient evaluations
EA3	138	0.058	1,242	138
GRG2	10	0.024	509	90
IQP	13	0.140	189	189
NAG8	12	0.040	252	243
RQP	22	0.035	666	198

Table 5. Computational effort for various test problems (see Table 2 of [38])

Problem	P1 ($n=2$)	P1 ($n=4$)	P2	P3	P4	P5	P6	P7	P8 ($n=4$)
Fastest	36	70	45	67	111	170	37	390	104
Slowest	272	620	156	600	1251	495	300	1680	472
Average	113	280	81	173	402	290	129	955	248
CG	56	139	49	134	130	212	23	93	184

VA08A, the Hatfield codes CONGRA and CONLS, a Fletcher–Reeves code with a new line search, the Buckley and Le Nir method, and four new methods OPCGI–OPCG4 presented in [38]. The stopping criterion was $|\mathbf{g}_k| \leqslant 0.01$, where the norm is the Euclidean norm, and the computational effort was measured by the expression $N_F + nN_G$ where n is the number of unknowns, N_F is the number of function calls, and N_G is the number of gradient calls. In Table 5 we list for each test problem, the lowest computational effort achieved by a code, the highest computational effort (among the codes that solved the test problem), the average computational effort corresponding to all the codes that solved a given test problem, and the computational effort for CG.

Observe that the computational effort associated with the Conjugate Gradient Search Scheme is well above the average and near the fastest algorithm in most cases, and in 2 cases it is the fastest algorithm. In [38] there were 3 algorithms that solved all the test problems of Table 5 successfully. If the computational effort for these 3 algorithms along with CG is averaged over all the test problems, we obtain the following averages; CONGRA (516), CONLS (496), VA08A (309), and CG (111). Hence, the Conjugate Gradient Search Scheme performs quite well relative to the most reliable algorithms.

Results for higher dimensional versions of the test problems seemed to parallel results in lower dimensions. For example, our algorithm did not fare as well on the Powell function (P8) as on the Rosenbrock function (P5). In extended versions of these functions (see [38], P8 and P9), the computational effort for CG is 4415 in P8 with $n = 60$ and 959 in P9 with $n = 20$. In [38] the computational efforts are 1688 (fastest), 9577 (slowest), and 4025 (average) for P8 and 801 (fastest), 4111 (slowest), and 1913 (average) for P9.

Some of the test problems in [39] were also examined. In [39] Shanno presents several different versions of the conjugate gradient method and compares them to each other as well as to a BFGS quasi-Newton method. In implementing the conjugate gradient schemes, he uses the same (relatively simple) search scheme so that the differences in efficiency are due to the conjugate gradient scheme and not to the line search efficiency. In comparing the conjugate gradient results to the BFGS quasi-Newton results, he concludes that in general quasi-Newton methods are superior to conjugate gradient methods. Note though that with a more involved search scheme such as the Conjugate Gradient Search Scheme, the conjugate gradient method may fare better. For example, solving Colville 4 with 4 different starting points, the BFGS method uses 42, 112, 111, and 51 function and gradient evaluations (see Table 4 in [39]). With CG the corresponding number of function (F) and gradient (G) evaluations are (53F, 22G), (72F, 29G), (623F, 239G), and (280F, 110G). Hence, in two of the cases, the quasi-Newton method seems superior while in two of the cases, the conjugate gradient method seems superior. For an extended Rosenbrock function with $n = 10$, the quasi-Newton method requires 993 function and gradient evaluations while CG requires 45 function and 21 gradient evaluations. In summary, the Conjugate Gradient Search Scheme seems promising.

REFERENCES

1. D. A. Pierre, *Optimization Theory with Applications*. Wiley, New York (1969).
2. G. R. Walsh, *Methods of Optimization*. Wiley, London (1975).
3. R. Mifflin, Stationarity and superlinear convergence of an algorithm for univariate locally Lipschitz constrained minimization. *Math. Program.* **28**, 50–71 (1984).
4. J. E. Dennis Jr and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, N.J. (1983).
5. M. Al-Baali and R. Fletcher, An efficient line search for nonlinear least squares. *J. Optim. Theory Applic.* **48**, 359–377 (1986).
6. P. E. Gill and W. Murray, Safeguarded steplength algorithms for optimization using descent methods. Report NAC 37, National Physics Laboratory, Teddington, England (1974).

7. N. Ghosh and W. W. Hager, A derivative-free bracketing scheme for univariate minimization.
8. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass. (1984).
9. J. J. Moré and D. C. Sorensen, Newton's method. Research Report 82-8, Argonne National Laboratory, Argonne, Ill. (1982).
10. W. C. Davidon, Variable metric method for minimization. Research Report 5990, Argonne National Laboratory, Lemont, Ill. (1959).
11. G. Dahlquist and A. Björck, *Numerical Methods* (Translated by N. Anderson). Prentice-Hall, Englewood Cliffs, N.J. (1974).
12. R. P. Brent, *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, N.J. (1973).
13. E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. Wiley, New York (1966).
14. A. Tamir, Rates of convergence of a one-dimensional search based on interpolating polynomials. *J. Optim. Theory Applic.* **27**, 187–203 (1979).
15. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York (1970).
16. A. Cohen, Rate of convergence for root finding and optimization algorithms. University of California at Berkeley, Ph.D. Thesis (1970).
17. J. W. Daniel, The conjugate gradient method for linear and nonlinear operator equations. *SIAM J. numer. Analysis* **4**, 10–26 (1967).
18. J. W. Daniel, Convergence of the conjugate gradient method with computationally convenient modifications. *Numer. Math.* **10**, 125–131 (1967).
19. J. W. Daniel, A correction concerning the convergence rate for the conjugate gradient method. *SIAM J. numer. Analysis* **7**, 277–280 (1970).
20. R. Klessig and E. Polak, Efficient implementations of the Polak–Ribiere conjugate gradient algorithm. *SIAM J. Control* **10**, 524–549 (1972).
21. M. L. Lenard, Practical convergence conditions for unconstrained minimization. *Math. Program.* **4**, 309–323 (1973).
22. M. L. Lenard, Convergence conditions for restarted conjugate gradient methods with inaccurate line searches. *Math. Program.* **10**, 32–51 (1976).
23. P. Baptis and J. Stoer, On the relation between quadratic termination and convergence properties of minimization algorithms, Part II, Applications. *Numer. Math.* **28**, 367–391 (1977).
24. J. Stoer, On the relation between quadratic termination and convergence properties of minimization algorithms, Part I. *Numer. Math.* **28**, 343–366 (1977).
25. M. Al-Baali, Descent property and global convergence of the Fletcher–Reeves method with inexact line search. *IMA J. numer. Analysis* **5**, 121–124 (1985).
26. J. G. Ecker and M. Kupferschmid, A computational comparison of the ellipsoid algorithm with several nonlinear programming algorithms. *SIAM J. Control Optim.* **23**, 657–674 (1985).
27. M. Kupferschmid and J. G. Ecker, EA3: A practical implementation of an ellipsoid algorithm for nonlinear programming. Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, N.Y. (1984).
28. L. S. Lasdon, A. Waren, A. Jain and M. W. Ratner, Design and testing of a generalized reduced gradient code for nonlinear programming. *ACM Trans. math. Software* **4**, 34–50 (1978).
29. S.-P. Han, A globally convergent method for nonlinear programming. *J. Optim. Theory Applic.* **22**, 297–309 (1977).
30. M. J. D. Powell, Algorithms for nonlinear constraints that use Lagrangian functions. *Math. Program.* **14**, 224–248 (1978).
31. R. L. Crane, K. E. Hillstrom and M. Minkoff, Solution of the general nonlinear programming problem with subroutine VMCON. Argonne National Laboratory, Report Number ANL-80-64, Argonne, Ill. (1980).
32. P. E. Gill and W. Murray, *Numerical Methods for Constrained Minimization*. Academic Press, New York (1974).
33. M. C. Biggs, Constrained minimization using recursive quadratic programming. In *Toward Global Optimization* (Ed. L. C. W. Dixon and G. P. Szëgo), pp. 341–349. North-Holland, Amsterdam (1975).
34. Hatfield Subroutine Library, *The Optima User's Manual*. Numerical Optimisation Centre, Hatfield Polytechnic Institute, Hatfield, Hertfordshire, U.K. (1976).
35. A. R. Colville, A comparative study of nonlinear programming codes. IBM New York Scientific Center Report Number 320-2949, 410 East 62nd Street, New York (1968).
36. R. Fletcher, *Practical Methods of Optimization*; Vol. 1, *Unconstrained Optimization*. Wiley, Chichester (1980).
37. W. W. Hager, *Applied Numerical Linear Algebra*. Prentice-Hall, Englewood Cliffs, N.J. (1988).
38. L. C. W. Dixon, P. G. Ducksbury and P. Singh, A new three-term conjugate gradient method. *J. Optim. Theory Applic.* **47**, 285–300 (1985).
39. D. F. Shanno, Conjugate gradient methods with inexact searches. *Math. Op. Res.* **3**, 244–256 (1978).
40. G. E. Forsythe, M. A. Malcom and C. B. Moler, *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs, N.J. (1977).

APPENDIX

The Local Minimum of a Cubic

Suppose that $p(x)$ is a cubic with values p_0 and p_1 at $x = 0$ and at $x = 1$ and with derivatives d_0 and d_1 at $x = 0$ and at $x = 1$. Then the derivative of p is given by

$$p'(x) = d_0 - 2(d_0 + v)x + (d_0 + d_1 + 2v)x^2, \tag{A.1}$$

where

$$v = d_0 + d_1 + 3(p_0 - p_1).$$

Except in the trivial case where p is constant, a cubic has a most one local minimum which is the root of the quadratic equation $p'(x) = 0$. Recall that two zeros x_+ and x_- of the quadratic $ax^2 + bx + c$ are

$$x_\pm = \frac{-b \pm (b^2 - 4ac)^{1/2}}{2a} \tag{A.2}$$

For the quadratic (A.1), we have $a = (d_0 + d_1 + 2v)$, $b = -2(d_0 + v)$, $c = d_0$, and

$$\frac{b^2 - 4ac}{4} = (d_0 + v)^2 - (d_0 + d_1 + 2v)d_0 = v^2 - d_0 d_1. \tag{A.3}$$

If the discriminant $b^2 - 4ac$ is less than zero, then $p'(x)$ never vanishes when x is real. Hence, p is strictly monotone and there is no local minimum. On the other hand, if $b^2 - 4ac$ is positive, then $p'(x)$ has two real zeros. Differentiating $p'(x)$ and inserting $x = x_\pm$, it can be verified that $p''(x_\pm) = \pm 2w$ where w is related to the discriminant:

$$w = (v^2 - d_0 d_1)^{1/2}.$$

Consequently, the local minimum of $p(x)$ occurs at $x = x_+$ where the second derivative is positive.

It is well known (see [40, pp. 20–21]) that if b is positive and $4|ac|$ is small relative to b^2, then the computed value of $-b + (b^2 - 4ac)^{1/2}$ is relatively inaccurate since nearly equal numbers are subtracted. Moreover, multiplying the numerator and the denominator of x_+ by $-b - (b^2 - 4ac)^{1/2}$, we obtain

$$x_+ = \frac{-2c}{b + (b^2 - 4ac)^{1/2}}, \tag{A.4}$$

which can now be evaluated accurately since the positive parameter b is added to the square root, not subtracted from the square root. In general, (A.4) yields a more accurate value for x_+ than (A.2) when b is positive and $b^2 > 4ac$. Differentiating (A.1) we see that $p''(0) = -2(d_0 + v) = b$. Thus the sign of b is the same as the sign of $p''(0)$. The second derivative of a real valued function f is typically positive in a neighborhood of a local minimum α. If f has two continuous derivatives, then the second derivative of a cubic interpolant will also be positive near α provided the interpolation points are sufficiently close to α. Furthermore, (A.3) implies that b^2 is greater than $4ac$ whenever $v^2 - d_0 d_1$ is positive or equivalently, whenever w is pure real. Hence, in a neighborhood of a local minimum, it is better to compute the local minimum of the cubic interpolant using relation (A.4) rather than relation (A.2). Substituting $a = (d_0 + d_1 + 2v)$, $b = -2(d_0 + v)$, $c = d_0$ in (A.4) gives

$$x_+ = \frac{d_0}{d_0 + v - w}. \tag{A.5}$$

In a similar manner, adding and subtracting one from (A.2), x_+ can be written

$$x_+ = 1 - \frac{d_1}{d_1 + v + w}, \tag{A.6}$$

and this formula is accurate when $p''(1) = 2(d_1 + v)$ is positive. After changing variables, (A.5) and (A.6) yield (4).