JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 62, 310–324 (1978)

A Global Minimization Algorithm for a Class of One-Dimensional Functions*

STEPHEN E. JACOBSEN AND MOHAMMED TORABI

Engineering Systems Department, School of Engineering and Applied Science, University of California, Los Angeles, California

Submitted by Masanao Aoki

An algorithm is developed for finding the global minimum of a continuously differentiable function on a compact interval in R_1. The function is assumed to be the sum of a convex and a concave function, each of which belongs to $C^1[a, b]$. Any one-dimensional function with a bounded second derivative can be so written and, therefore, such functions generally have many local minima. The algorithm utilizes the structure of the objective to produce an ϵ-optimal solution by a sequence of simple one-dimensional convex programs.

1. INTRODUCTION

We consider the problem, denoted by (P),

$$\begin{aligned} &\min f(x) \\ &x \in [a, b] \end{aligned} \tag{P}$$

where $f = h + g$, h and g are convex and concave respectively, and both h and g are differentiable on an open interval containing $[a, b]$. The latter assumption, together with the fact that h and $-g$ are convex, implies that h' and g' are both continuous. Note that any function f with a bounded second derivative is of this class. That is, let $M > 0$ be chosen so that

$$2M \geqslant \sup\{f''(x) \mid x \in [a, b]\}$$

and rewrite f as

$$f(x) = Mx^2 + (f(x) - Mx^2).$$

Therefore, such functions may have many local minima and, as a result, the application of methods which require the function to be unimodal will lead, at best, to a local minimum and not necessarily to a global minimum.

* This research was supported by the National Science Foundation, Grant ENG 74-02629.

0022-247X/78/0622-0310$02.00/0

This paper develops an algorithm for finding an ϵ-optimal solution in a finite number of steps. That is, if x^{**} is a global minimizer, we seek an x^* so that $f(x^{**}) \geqslant f(x^*) - \epsilon$. The algorithm utilizes the structure of f and locates an ϵ-optimal solution by a sequence of one-dimensional convex programs. Therefore, efficient unimodal methods [1, 2] can be incorporated as subroutines. The algorithm does make explicit use of h' and g'.

Recently, Shubert [3] has developed a sequential search method for optimizing a one-dimensional function on a compact interval. The only requirement is that the function be globally Lipschitzian and that the Lipschitz constant be known. He demonstrates that his sampling rule is minimax with respect to the class of functions with the same Lipschitz constant.

Also, Brent [4] has developed a method for optimizing a one-dimensional function, on a compact interval, with a bounded second derivative. His procedure does not require the computation of first derivatives but does require knowledge of a small upper bound for the second derivative.

The purpose of this paper is to demonstrate how rather simple gradient methods can also be used to find a global minimum for one-dimensional function on a compact interval.

The algorithm is coded in Fortran IV for use on the 360/91 and the last section of the paper presents some examples.

Section 2 contains the results which motivate (in particular, Theorems 3 and 4) the algorithm of Section 3. Figure 1 of Section 3 is useful for the geometric interpretation of the results of Section 2.

2. Main Results

THEOREM 1. *Let $u_i \in [a, b)$ and let u_{i+1} solve the convex program $L_g(u_i)$*

$$\begin{aligned} &\min h(x) + g'(u_i)\, x \\ &x \in [u_i\,,\, b] \end{aligned} \qquad (L_g(u_i))$$

Then u_{i+1} solves

$$\begin{aligned} &\min f(x) \\ &x \in [u_i\,,\, u_{i+1}]. \end{aligned}$$

Proof. By optimality of u_{i+1} for $L_g(u_i)$ and by concavity of g we have

$$h(u_i) \geqslant h(u_{i+1}) + g'(u_i)\,(u_{i+1} - u_i) \geqslant h(u_{i+1}) + g(u_{i+1}) - g(u_i).$$

Therefore, $f(u_i) \geqslant f(u_{i+1})$. Now, assume there is a $\bar{u} \in (u_i\,,\, u_{i+1})$ so that $f(\bar{u}) < f(u_{i+1})$. By concavity of g we have

$$\begin{aligned} g'(u_i)\,(\bar{u} - u_{i+1}) &\leqslant g'(\bar{u})\,(\bar{u} - u_{i+1}), \\ g(u_{i+1}) &\leqslant g(\bar{u}) + g'(\bar{u})\,(u_{i+1} - \bar{u}), \end{aligned}$$

and, by assumption,

$$h(\bar{u}) + g(\bar{u}) < h(u_{i+1}) + g(u_{i+1}).$$

Adding these three inequalities we get

$$h(\bar{u}) + g'(u_i)\,\bar{u} < h(u_{i+1}) + g'(u_i)\,u_{i+1}$$

which contradicts the optimality of u_{i+1} for $L_g(u_i)$.

Note that Theorem 1 replaces f with a convex approximation

$$f_1(x; u_i) = h(x) + g(u_i) + g'(u_i)\,(x - u_i)$$

and, of course, $f_1(x; u_i) \geqslant f(x)$ on $[a, b]$ and $f_1'(x; u_i) \geqslant f'(x)$ on $[u_i, b]$ because of concavity of g.

We now present a convergence result on the iterative solution of problems $L_g(u_i)$).

THEOREM 2. *Let $u_1 \in [a, b)$ be such that $f'(u_1) < 0$. Define, for $i = 1, 2, \ldots$,*

$$u_{i+1} = \{\min x \mid x \in [u_i, b],\ h(x) + g'(u_i)\,x \leqslant \min_{y \in [u_i, b]} h(y) + g'(u_i)\,y\}.$$

Then $u_i \to \bar{u}$ where $f'(u) < 0$ for all $u \in [u_1, \bar{u})$ and $f'(\bar{u}) = 0$ if $u_i < b$ all i. In addition, $\bar{u}$ solves

$$\min f(x)$$
$$x \in [u_1, \bar{u}].$$

Proof. First observe that if $f'(u_i) \geqslant 0$ then $u_{i+1} = u_i$ is an optimal solution for $L_g(u_i)$. Also, if $f'(u_i) < 0$ then any optimal solution, u_{i+1}, for $L_g(u_i)$ is such that $u_i < u_{i+1}$. These facts follow by convexity and the fact that $f'(u_i)$ is the value of the derivative, at u_i, of the objective function of $L_g(u_i)$. Also note that if $f'(u_i) \leqslant 0$ then $f'(u) \leqslant 0$ for all $u \in [u_i, u_{i+1}]$. This follows since the optimality of u_{i+1} for $L_g(u_i)$ together with the concavity of g and convexity of h imply

$$0 \geqslant h'(u_{i+1}) + g'(u_i) \geqslant h'(u_{i+1}) + g'(u) \geqslant h'(u) + g'(u) = f'(u).$$

Now, suppose at some iteration, i, $f'(u_i) = 0$ and i is the first such iteration. Then $u_{i+1} = u_i$ and set $\bar{u} = u_i$. Also, suppose at some iteration, i, $f'(u_i) < 0$ and $u_{i+1} = b$. Then set $\bar{u} = b$ and by the above we have $f'(\bar{u}) \leqslant 0$. Therefore, assume $f'(u_i) < 0$ and $u_{i+1} < b$ for all i. By boundedness and monotonicity both sequences $\{u_i\}$ and $\{u_{i+1}\}$ converge to the same limit, $\bar{u}$, and by continuity of both h' and g' we have $f'(\bar{u}) = 0$. Iterative use of Theorem 1 implies that $\bar{u}$ minimizes f on the interval $[u_1, \bar{u}]$.

Now let $y_j \in [a, b)$ and define the function, on an open interval containing $[0, 1]$,

$$\sigma_j(\lambda) = h(y_j + \lambda(b - y_j)) + \lambda(g(b) - g(y_j)) + g(y_j).$$

Then σ_j is convex on $[0, 1]$, $\sigma_j(0) = f(y_j)$, $\sigma_j(1) = f(b)$, and $\sigma_j(\lambda) \leqslant f(y_j + \lambda(b - y_j))$ for $\lambda \in [0, 1]$. That is, by setting $x = y_j + \lambda(b - y_j)$, we have that

$$\sigma_j\left(\frac{x - y_j}{b - y_j}\right) = h(x) + \frac{g(b) - g(y_j)}{b - y_j}(x - y_j) + g(y_j) \leqslant f(x)$$

for $x \in [y_j, b]$. We then have the following.

THEOREM 3. *Let $y_j \in [a, b)$ and let UB be some upper bound for problem* (P) *such that $f(y_j) \geqslant UB$. Consider the convex program, $S(y_j)$,*

$$\text{subject to} \qquad \min_{\lambda \geqslant 0} \lambda \quad \sigma_j(\lambda) \leqslant UB \qquad S(y_j)$$

Then

(i) *If $\lambda_j \leqslant 1$ is optimal for $S(y_j)$, the interval $[y_j, y_j + \lambda_j(b - y_j)]$ contains no points x such that $f(x) < UB$.*

(ii) *If $\lambda_j > 1$ is optimal for $S(y_j)$ or if $S(y_j)$ is infeasible, then the interval $[y_j, b]$ contains no points x such that $f(x) < UB$.*

Proof.

(i) Note that if $f(y_j) = UB$ then $\lambda_j = 0$ is optimal. If $f(y_j) > UB$ then $\lambda = 0$ is infeasible for $S(y_j)$. Now, assume $\lambda_j > 0$. Then $\lambda \in [0, \lambda_j)$ is infeasible for $S(y_j)$ and

$$\sigma_j(\lambda) \leqslant f(y_j + \lambda(b - y_j))$$

for all $\lambda \in [0, \lambda_j]$. Therefore, $f(y_j + \lambda(b - y_j)) \geqslant UB$ for all $\lambda \in [0, \lambda_j]$.

(ii) If $\lambda_j > 1$ or if $S(y_j)$ is infeasible, then for all $\lambda \in [0, 1]$ we have

$$UB < \sigma_j(\lambda) \leqslant f(y_j + \lambda(b - y_j)).$$

Note that if $\sigma_j'(0) \geqslant 0$ and $f(y_j) > UB$, then $S(y_j)$ has no feasible solution in $[0, 1]$.

As in Theorem 2, we can iteratively solve problems $S(y_j)$. That is, if $\lambda_j < 1$ is the optimal solution to $S(y_j)$ and $y_{j+1} = y_j + \lambda_j(b - y_j)$ is such that $f(y_{j+1}) > UB$ then solve $S(y_{j+1})$.

THEOREM 4. *Let $y_1 \in [a, b)$ be such that $f(y_1) > UB$. Assume for all j that the optimal solution for $S(y_j)$ is $\lambda_j < 1$. Then, the sequence $\{y_j\}$ converges to a point $\bar{y}$ such that $f(\bar{y}) = UB$. In addition, for all $x \in [y_1, \bar{y})$ we have $f(x) > UB$.*

Proof. Suppose at iteration j we have $f(y_{j+1}) = UB$ and j is the first such iteration; therefore, because of Theorem 2, $f(y_\nu) > UB$ for all $\nu = 1,\ldots, j$. Then $\lambda_{j+1} = 0$ and we can set $\bar{y} = y_{j+1}$. Therefore, the only case left to consider is $f(y_j) > UB$ for all j. Then $\lambda_j \in (0, 1)$ for all j and hence, for all j, $y_j < y_{j+1} < b$ and $h(y_{j+1}) + \lambda_j(g(b) - g(y_j)) + g(y_j) = UB$. By monotonicity and boundedness, both sequences $\{y_j\}$ and $\{y_{j+1}\}$ converge to the same limit, $\bar{y}$, and therefore $\lambda_j(g(b) - g(y_j)) \to 0$. This latter claim is true since if $g(y_j)$ does not converge to $g(b)$, then $\bar{y} < b$ and therefore $y_{j+1} = y_j + \lambda_j(b - y_j)$, for all j, implies $\lambda_j \to 0$. Hence,

$$UB = \lim_{j\to\infty}(h(y_{j+1}) + \lambda_j(g(b) - g(y_j)) + g(y_j)) = f(\bar{y}).$$

Iterative use of Theorem 3 implies $f(x) \geqslant UB$ for all $x \in [y_1, \bar{y}]$.

Note that Theorems 3 and 4 imply that a sequence of $S(y_j)$ problems can be used to move from a point $x_k = y_1$, $f(x_k) > UB_k$, to a point, x_{k+1}, (if one exists) such that $f(x_{k+1}) = UB_k$ and there are no better points in the interval $[x_k, x_{k+1}]$. In fact, these theorems form the basis of the algorithm.

3. The Algorithm

In this section we present the major steps of an algorithm which is suggested by Theorems 1–4. We defer discussion of the details to a later section. See Fig. 1 for the geometric idea of the procedure.

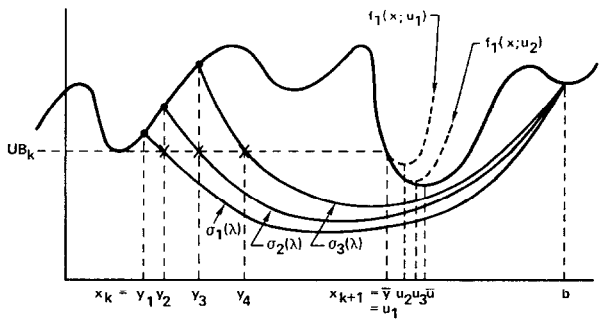

FIG. 1. Geometric interpretation of the algorithm.

Let UB be some initial upper bound for (P). For instance, $UB = f(a)$, $UB = \min\{f(a), f(b)\}$, or UB could be the best value of the objective f found by, say, some coarse grid or random search method.

Step 0. Set $k = 0$, $x_k = a$, $UB_k = UB$; go to Step 1.

Step 1.

If $f(x_k) > UB_k$, find out whether or not there exists $x_{k+1} \in (x_k, b]$ such that $f(x_{k+1}) = UB_k$.

(Recall, Theorems 3 and 4 imply that such a point, if it exists, can be found by a sequence of $S(y_j)$ problems, where $y_1 = x_k$)

If such an x_{k+1} exists, set $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 2. If such an x_{k+1} does not exist, then any $x^* \in \{x \in [a, b] \mid f(x) \leqslant UB_k\}$ is optimal. If $f(x_k) \leqslant UB_k$, set $UB_k = f(x_k)$ and go to Step 2.

Step 2.

If $f'(x_k) < 0$, find out whether or not there exists $x_{k+1} \in (x_k, b]$ such that $f'(x_{k+1}) = 0$ and $f(x_{k+1}) < UB_k$.

(Recall, Theorems 1 and 2 imply that such a point, if it exists, can be found by a sequence of $L_g(u_i)$ problems, where $u_1 = x_k$)

If such an x_{k+1} exists, set $UB_{k+1} = f(x_{k+1})$, $k = k + 1$, and go to Step 3.

If such an x_{k+1} does not exist, then $x^* = b$ is optimal. If $f'(x_k) \geqslant 0$, go to Step 3.

Step 3. Increment x_k slightly to a point $x_{k+1} > x_k$, set $UB_{k+1} = \min\{UB_k, f(x_{k+1})\}$, $k = k + 1$, and go to Step 1.

In the next section we discuss appropriate modifications and numerical aspects of the basic algorithm. The appendix contains a more detailed, and typical, algorithm needed for machine implementation.

4. Discussion and Modifications of the Algorithm

Note that Steps 1 and 2 of the algorithm involve possibly infinite sequences of simple convex problems each of which may also involve an infinite number of steps. Therefore, it is clear that tolerance parameters must be introduced so the procedure cannot cycle within a major step.

Problem $S(y_j)$ is easily solved by, say, Newton's procedure beginning at $\lambda = 0$, $\sigma_j(0) > UB_k$. Therefore, if there is a root for $\sigma_j(\lambda) = UB_k$ then, within a finite number of steps, a point λ_j must be reached such that λ_j is ϵ_1-optimal. That is, we say λ_j is ϵ_1-optimal if

$$UB_k \leqslant \sigma_j(\lambda_j) \leqslant UB_k + \epsilon_1 .$$

Therefore, if we choose to stop solving $S(y_j)$ problems when we reach a point $\bar{y}$ such that

$$UB_k \leqslant f(\bar{y}) \leqslant UB_k + \Delta,$$

where $\Delta > \epsilon_1$, then we will solve at most a finite number of $S(y_j)$ problems at Step 1. This is true since $\sigma_j(\lambda_j) \leqslant f(y_{j+1})$ and if $f(y_{j+1}) \leqslant UB_k + \epsilon_1$, then Step 1 is exited since $\Delta > \epsilon_1$. Therefore, if $\lambda_j \in (0, 1)$ is ϵ_1-optimal for $S(y_j)$ and $f(y_{j+1}) > UB_k + \epsilon_1$, all j, then the proof of Theorem 4 shows $f(y_j) \to UB_k + \epsilon_1$ and, therefore, within a finite number of steps, a point $\bar{y}$ must be reached such that $UB_k \leqslant f(\bar{y}) \leqslant UB_k + \Delta$.

Similarly, we now show that, by the introduction of appropriate tolerance parameters, we will solve at most a finite number of $L_g(u_i)$ problems at Step 2 of the algorithm. We say u_{i+1} is ϵ_2-optimal for $L_g(u_i)$ if u_{i+1} is optimal or

$$\epsilon_2 \geqslant h'(u_{i+1}) + g'(u_i) \geqslant -\epsilon_2 .$$

Therefore, if we choose to stop solving $L_g(u_i)$ problems when we reach a point $\bar{u}$ such that $f'(\bar{u}) \geqslant -\epsilon_3$, $\epsilon_3 > \epsilon_2$, then we will solve at most a finite number of $L_g(u_i)$ problems at Step 2. To see this, if for all i we have $f'(u_i) < -\epsilon_2$ and $h'(u_{i+1}) + g'(u_i) \geqslant -\epsilon_2$, then, as in the proof of Theorem 2, we have that $f'(u_i) \to -\epsilon_2$ and therefore, within a finite number of steps, a point $\bar{u}$ must be reached such that $f'(\bar{u}) \geqslant -\epsilon_3$ (recall, $\epsilon_3 > \epsilon_2$). Each $L_g(u_i)$ problem is easily solved by, say, some unimodal search method or, in some cases, by using Newton's procedure for finding a root of $h'(u) = -g'(u_i)$. However, care must be taken in the latter case since h' is not generally convex.

Another aspect of Step 1 requires some discussion. Even if $f(x_k) > UB_k$, it may be the case that the solution of $S(y_1)$, $y_1 = x_k$, leads to a point y_2 which is not much to the right of y_1. For instance, suppose $S(y_1)$ is solved by Newton's method starting at $\lambda = 0$. The first step of this procedure moves us from $\lambda = 0$ to

$$\lambda = \frac{UB_k - f(y_1)}{\sigma_1'(0)},$$

where $\sigma_1'(0) = (b - y_1)\, h'(y_1) + g(b) - g(y_1)$. Therefore, we see that if $f(y_1)$ is very close (and larger) than UB_k or if $\sigma_1'(0)$ is relatively large in absolute value, then this step may be quite small and, therefore, the iterative procedure suggested by Theorems 3 and 4 may converge slowly. When $f'(x_k) < 0$ and the step is too small, it is desirable to go to Step 2 of the basic algorithm. This is the case for the algorithm contained in the appendix. When $f'(x_k) > 0$ and the step is too small, the following results are useful.

Basically, the idea is to move "uphill" to a point v such that $f(v) - UB_k$ is relatively large and such that there exists no point $x \in [y_1, v]$ such that $f(x) < f(y_1)$. Of course, the idea is to obtain a relatively larger step to the right

(from y_1) by this procedure. Suppose we approximate, at $v_1 = x_k$, the function f by the concave function

$$f_2(x; v_1) = g(x) + h(v_1) + h'(v_1)(x - v_1).$$

Then $f_2(x; v_1) \leqslant f(x)$ and therefore if we can find the largest x such that $f_2(x; v_1) \geqslant f(v_1)$, then (if v_2 is such an x) there does not exist any point $x \in [v_1, v_2]$ such that $f(x) < f(v_1)$ and, of course, $f(v_2) \geqslant f(v_1)$. For convenience, let

$$R_j(x) = f_2(x; v_j) - f(v_j).$$

We restate the above as a theorem.

THEOREM 5. *Let $v_j \in [a, b)$ and let v_{j+1} solve the concave program*

$$\max x$$

subject to

$$R_j(x) \geqslant 0 \qquad (L_h(v_j))$$

$$x \in [v_j, b]$$

where $R_j(x) = g(x) - g(v_j) + h'(v_j)(x - v_j)$. Then the interval $[v_j, v_{j+1}]$ contains no points x such that $f(x) < f(v_j)$.

Before continuing observe that if $f'(v_j) > 0$ then the optimal solution, v_{j+1}, for $L_h(v_j)$ is such that $v_{j+1} > v_j$. This follows since $R_j(v_j) = 0$, $R_j'(v_j) = f'(v_j)$, and therefore a slight increase to the right of v_j will maintain feasibility of the constraints for $L_h(v_j)$.

We also have the following result on iterative solutions of problems $L_h(v_j)$.

THEOREM 6. *Let $v_1 \in [a, b)$ be such that $f'(v_1) > 0$. Define, for $j = 1, 2, \ldots$,*

$$v_{j+1} = \max\{x \mid x \in [v_j, b], R_j(x) \geqslant 0\}.$$

Then $v_j \to \bar{v}$, and if $\bar{v} < b$ then $f'(\bar{v}) \leqslant 0$. In addition, there does not exist any $x \in [v_1, \bar{v}]$ so that $f(x) < f(v_1)$.

Proof. If at some iteration, j, $v_j = b$ then $v_{j+1} = b$ and then set $\bar{v} = b$. Also, if for some j we have $0 > f'(v_j) = R_j'(v_j)$, then concavity of R_j together with $R_j(v_j) = 0$ imply $v_{j+1} = v_j$ and therefore we can set $\bar{v} = v_j$. Therefore, assume for all j that $f'(v_j) \geqslant 0$ and $v_j < b$. Then concavity of R_j together with $R_j(v_j) = R_j(v_{j+1}) = 0$ and $R_j'(v_j) = f'(v_j)$ imply $0 \geqslant R_j'(v_{j+1}) = g'(v_{j+1}) + h'(v_j)$ for all j. By boundedness and monotonicity, both sequences $\{v_j\}$ and $\{v_{j+1}\}$ converge to the same limit, $\bar{v}$, and by continuity of both g' and h' we have that $f'(\bar{v}) \leqslant 0$. Iterative use of Theorem 5 implies $f(v) \geqslant f(v_1)$ for all $v \in [v_1, \bar{v}]$.

Theorems 5 and 6 imply that we can, by iterative use of problems $L_h(v_j)$, possibly "escape" from a point y_1, such that $f'(y_1) > 0$, $f(y_1) > UB_k$, and

the optimal solution, y_2, for $S(y_1)$ is not much to the right of y_1. As noted above, such may occur if

$$\frac{UB_k - f(y_1)}{\sigma_1'(0)}$$

is small. Note that there is no need to necessarily solve many $L_h(v_j)$ problems. For instance, we may stop at some v_j, and go to Step 1, if

$$\frac{UB_k - f(v_j)}{\sigma_j'(0)}$$

is sufficiently large. This is the procedure adopted in the algorithm of the appendix. In fact, we agree to cease solving the $L_h(v_j)$ problems (see Step 5b of appendix algorithm) if for some v_j the step-size above is sufficiently large or if for some v_j the step-size is small and $f'(v_j) \leqslant \epsilon_3$.

As for the $L_g(u_i)$ and $S(y_j)$ problems, we need to show that we will solve at most a finite number of $L_h(v_j)$ problems. In particular, $L_h(v_r)$ is also easily solved by Newton's method starting at $v = b$, $R_r(b) < 0$ (note that if $R_r(b) \geqslant 0$, then an optimizer for $L_h(v_r)$ is at least equal to b). Therefore, if $R_r(b) < 0$ and $R_r'(v_r) = f'(v_r) > 0$ there is a root for $R_r(v) = 0$ in (v_r, b). Then, within a finite number of steps, we must reach a point, v_{r+1}, such that $-\epsilon_4 \leqslant R_r(v_{r+1}) \leqslant 0 = R_r(v_r)$. Hence, if $f'(v_r) > 0$ and $v_r < b$, all r, the concavity of R_r implies $0 \geqslant R_r'(v_{r+1}) = g'(v_{r+1}) + h'(v_r)$, all r, and as in the proof of Theorem 6 we have that within a finite number of steps a point, $\bar{v}$, must be reached such that $f'(\bar{v}) \leqslant \epsilon_3$.

Before continuing, we observe that the solutions of problems $L_g(u_i)$ and $S(y_j)$ are simplified when we use the representation $\bar{f}(x) = Mx^2 + (f(x) - Mx^2)$. In this case, since $L_g(u_i)$ has a quadratic objective, we have that

$$u_{i+1} = \min\{b, u_i - f'(u_i)/2M\}.$$

Similarly, the problem of finding the smallest root (if real roots exist) of $\sigma_j(\lambda) = UB_k$ reduces to finding the roots of a quadratic equation.

Therefore, when this representation is used, we may set $\epsilon_1 = \epsilon_2 = 0$. Of course, then, when this representation is used the above step-size test becomes "is $y_{j+1} - y_j$ sufficiently large?" where y_{j+1} is the smallest root (if real roots exist) of

$$\sigma_j\left(\frac{x - y_j}{b - y_j}\right) = UB_k.$$

5. Perturbation and ϵ-Optimality

In the last section we discussed the introduction of tolerance parameters into the various subproblems. Therefore, in general, a sequence of $L_g(u_i)$ problems

will be stopped at a point $\bar{u}$ such that $0 \geqslant f'(\bar{u}) \geqslant -\epsilon_3$. Therefore, in order to now initiate a sequence of $S(y_j)$ problems (or, possibly, a sequence of $L_h(v_j)$ problems), or possibly another sequence of $L_g(u_i)$ problems, a perturbation step of size, say, δ is required. In fact, the procedure in the appendix perturbs when $|f'(x_k)| \leqslant \epsilon_3$. Therefore, some examination of the effect of such perturbations is in order.

Let f'' exist and let M_2 be an upper bound for f'' on $[a, b]$. Let $x_k \in [a, b)$ and let $x_{k+1} = x_k + \delta$. Now, for any $x \in [x_k, x_{k+1}]$ we have $f(x) - f(x_k) = f'(\xi)(x - x_k)$, where ξ is some point between x_k and x_{k+1}. But

$$|f'(\xi) - f'(x_k)| \leqslant M_2\delta$$

and therefore,

$$f(x) - f(x_k) \geqslant (f'(x_k) - M_2\delta)(x - x_k).$$

Now, if $f'(x_k) \geqslant -\epsilon_3$, we have

$$f(x) - f(x_k) \geqslant -(\epsilon_3 + M_2\delta)(x - x_k) \geqslant -(\epsilon_3 + M_2\delta)\delta.$$

Therefore, if we want a perturbation to have the property that $f(x) - f(x_k) \geqslant -\epsilon$, it suffices to choose a δ, given ϵ_3, such that

$$(\epsilon_3 + M_2\delta)\delta \leqslant \epsilon.$$

This means that if we choose

$$\delta = \frac{-\epsilon_3 + (\epsilon_3^2 + 4M_2\epsilon)^{1/2}}{2M_2}$$

then, given ϵ_3, a perturbation of size δ from x_k to $x_{k+1} = x_k + \delta$ will have the property that $f(x) - f(x_k) \geqslant -\epsilon$ for all $x \in [x_k, x_{k+1}]$ and, since $f(x_k) \geqslant UB_k$, we also have $f(x) \geqslant UB_k - \epsilon$.

Therefore, if for a particular problem M_2 is known or can easily be found, then there is no problem in choosing a perturbation parameter δ so as to guarantee $f(x) - f(x_k) \geqslant -\epsilon$ for $x \in [x_k, x_k + \delta]$ when $f'(x_k) \geqslant -\epsilon_3$. On the other hand, if M_2 is unknown then we must generally *assume* that the perturbation parameter is small enough so that perturbation cannot cause us to overlook minima whose objective values are significantly less than the current bound UB_k.

We now discuss ϵ-optimality in the context of the algorithm of the appendix. Note that the algorithm is not "complete" in that the procedures for solving the $L_g(u_i)$, $S(y_j)$, and $L_h(v_r)$ subproblems are not specified. Note also that the algorithm is the basic algorithm of Section 3 with (i) the step-size test for the $S(y_j)$ problems (Step 2), (ii) the "hill climbing" routine if the step-size test fails (Step 5), and (iii) the appropriate tolerance parameters as discussed in Section 4. Step 6 is the perturbation step.

Let a *mode* for f on $[a, b]$ be defined as follows. A subinterval $[x, y]$ of $[a, b]$ is said to be a *mode* if $f'(z) = 0$ on $[x, y]$.

Inspection of the algorithm (of the appendix) shows that Step 7 must be reached if f has a finite number of modes on $[a, b]$. In particular, Step 6 (perturbation) guarantees that nonsingleton modes (i.e., $x < y$) can cause no problem.

It only remains to discuss the accuracy of x^* at Step 7 of the algorithm. Step 7, or termination, is entered through Step 2 or Step 4 or Step 5 or Step 6. Step 7 can be entered via Step 2 only if for some $S(y_j)$ problem ($y_1 = x_k$) we have that $\lambda_j \geqslant 1$ or $S(y_j)$ is infeasible. In this case, the accuracy of x^* is associated only with the accuracy of UB_k. Step 7 can be entered via Step 4 only if b is optimal for some $L_g(u_i)$ problem ($u_1 = x_k$). If $f(b)$ is considerably smaller than UB_k, then $x^* = b$ is optimal. If $f(b)$ is considerably larger than UB_k, then accuracy of x^* is associated only with the accuracy of UB_k. The same is true if $f(b)$ is equal (or nearly equal) to UB_k. Step 7 can be entered via Step 5 only if b is optimal for some $L_h(v_r)$ problem, where $v_1 = x_k$. In this case the accuracy of x^* is again only associated with the accuracy of UB_k. Therefore, if the perturbation parameter δ is appropriately chosen (as discussed above) then the algorithm finds an ϵ-optimal solution. That is, if x^{**} is a global minimizer, the algorithm produces an x^* such that $f(x^{**}) \geqslant f(x^*) - \epsilon$.

6. Numerical Examples

The algorithm was coded in Fortran IV for use on the IBM 360/91. We now state the results of some problems.

Problem 1. Let $a = 0$, $b = 7.5$, $h(x) = x^4 + 41x^2$, $g(x) = 10(x - 8)^{-5} - 12x^3 - 18(x + 4)$. The function is sketched in Fig. 2.

For this problem we chose $\Delta = 5$, $\delta_1 = 0.005$, $\delta = 0.05$, $\epsilon_1 = 0.0005$, $\epsilon_2 = 0.0005$, $\epsilon_3 = 0.05$, $\epsilon_4 = 0.01$, $UB = f(a)$.

The algorithm starts from $x_1 = 0$ and after four L_4 problems stops at $x_2 = 0.2449515$. The algorithm then perturbs to $x_3 = 0.2949515$ and after solving three L_h problems stops at $x_4 = 1.8360970$. The algorithm then initiates, at this latter point, a sequence of 53 $S(y_j)$ problems and then stops at $x_5 = 7.482527$. The algorithm then initiates, at x_5, one L_g problem and concludes that $x_6 = 7.5$ is optimal with $f(x_6) = -119.1875$. The execution time for this problem is 0.31 seconds on the 360/91. Note that the algorithm bypassed the second local minimum when moving from x_4 to x_5. Also, each of the subproblems $L_g(u_i)$, $S(y_j)$, and $L_h(v_r)$ was solved by Newton's method.

Problem 2. We altered the above problem by changing b from 7.5 to 7.2. Thus the optimum is at the first local minimum (see Fig. 2). Also, we chose the parameter values $\Delta = 5$, $\delta_1 = 0.005$, $\delta = 0.005$, $\epsilon_1 = 0.0005$, $\epsilon_2 = 0.0005$, $\epsilon_3 = 0.005$, $\epsilon_4 = 0.01$, $UB = f(a)$. The sequence of steps is as follows. The

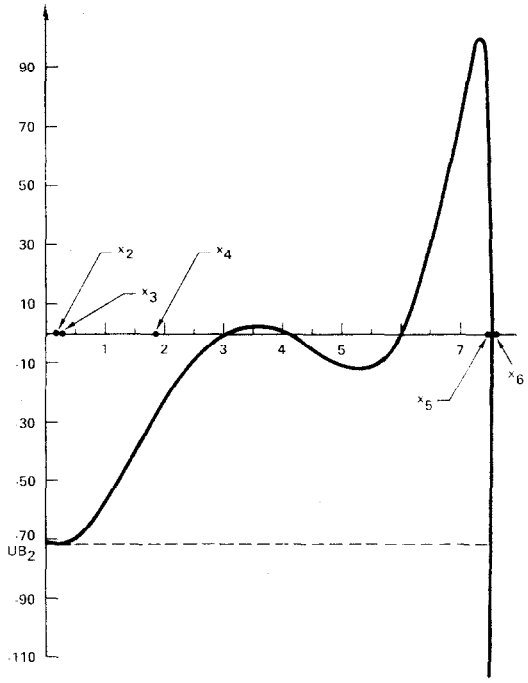

FIG. 2. Sketch of f and sequence of points x_k for minimizing f on [0, 7.5].

algorithm initiates, at $x_1 = 0$, five L_g problems and stops at $x_2 = 0.2451384$. The algorithm then perturbs to $x_3 = 0.2501384$ and initiates, at x_3, four L_h problems and stops at $x_4 = 1.771981$. The algorithm then initiates, at x_4, 36 $S(y_j)$ problems (starting from $y_1 = x_4$ to $y_{36} = 6.084911$). The algorithm then concludes that $S(y_{36})$ is infeasible and, therefore, $x^* = x_2$ is ϵ-optimal with $f(x_2) = -74.12217$. Note that we do not actually know ϵ since we have not computed M_2 for this problem. However, since we see (graphically) that f is convex on $[x_2, x_3]$, we have, for $x \in [x_2, x_3]$,

$$f(x) - f(x_2) \geqslant f'(x_2)(x - x_2) \geqslant -\delta\epsilon_3 .$$

That is, x_2 is $\delta\epsilon_3$-optimal where $\delta\epsilon_3 = 0.000025$. The execution time is 0.25 sec.

Problem 3. Same as Problem 1 and the problem is run in the form $\bar{f}(x) = (M_2/2)x^2 + (f(x) - (M_2/2)x^2)$ with $M_2 = 106$ (the upper bound for f'' was found by a fine grid search for the purpose of this example; of course, such information cannot be considered as generally available). Therefore, we can set $\epsilon_1 = \epsilon_2 = 0$. Also, to help see the effect of a good initial bound, we chose $UB = f(b)$, the optimal objective value (the remaining parameters are those of Problem 1).

The algorithm starts at $x_1 = 0$ with 22 $S(y_j)$ problems, $y_1 = x_1$, and stops at $y_{22} = 7.499999$. The algorithm then initiates, at $u_1 = y_{22}$, one L_g problem and concludes that $x^* = b$ is optimal. The execution time is 0.05 sec.

7. Appendix

This section contains a more detailed version of the algorithm or procedure of Section 3. The algorithm is not necessarily in the most "economical" form for there are various modifications of the basic algorithm that one can think of. The algorithm is essentially that of Section 3 with step-size test (for $S(y_j)$) and the "hill climbing" routine included. Also, note that if the representation $\bar{f}(x) = Mx^2 + (f(x) - Mx^2)$ is to be used then, as mentioned in Section 4, we can set $\epsilon_1 = \epsilon_2 = 0$. Also, in this case, Step 2 would be slightly modified as follows. We would actually solve $S(y_j)$ and if $y_{j+1} < b$ is optimal we would then ask if $y_{j+1} - y_j \geqslant \delta_1$. If not, we would go to Step 3. If yes, we would solve $S(y_{j+1})$ if $f(y_{j+1}) > UB_k + \Delta$; otherwise, we would go to Step 3.

As in Section 3, UB is some initial upper bound for problem (P).

Step 0. Select $\epsilon > 0$, $\epsilon_1 > 0$, $\epsilon_2 > 0$, $\epsilon_3 > \epsilon_2$, $\epsilon_4 > 0$, $\delta_1 > 0$, $\Delta > \epsilon_1$, $\delta > 0$. Set $k = 1$, $UB_{k-1} = UB$, $x_k = a$, and go to Step 1.

Step 1. Compute $f(x_k)$ and set $UB_k = \min\{UB_{k-1}, f(x_k)\}$.

If $x_k = b$, go to Step 7.

If $x_k < b$ and $UB_k \leqslant f(x_k) \leqslant UB_k + \Delta$, go to Step 3.

If $x_k < b$ and $UB_k + \Delta < f(x_k)$, go to Step 2.

Step 2. Set $j = 1$, $y_j = x_k$, and go to Step 2a.

Step 2a. Compute $\sigma_j'(0) = (b - y_j)\, h'(y_j) + g(b) - g(y_j)$.

If $\sigma_j'(0) \geqslant 0$, go to Step 7.

If $\sigma_j'(0) < 0$, go to Step 2b.

Step 2b.

If $[UB_k - f(y_j)]/\sigma_j'(0) \geqslant \delta_1$, go to Step 2c.

If $[UB_k - f(y_j)]/\sigma_j'(0) < \delta_1$, go to Step 3.

Step 2c. Solve $S(y_j)$.

If $S(y_j)$ is infeasible or if the optimal solution, λ_j, is such that $\lambda_j > 1$, go to Step 7.

If $\lambda_j \in (0, 1)$ is ϵ_1-optimal for $S(y_1)$ (e.g., $UB_k \leqslant \sigma_j(\lambda_j) \leqslant UB_k + \epsilon_1$) and $f(y_j + \lambda_j(b - y_j)) > UB_k + \Delta$, go to Step 2d.

If $\lambda_j \in (0, 1)$ is ϵ_1-optimal for $S(y_j)$ and $UB_k \leqslant f(y_j + \lambda_j(b - y_j)) \leqslant UB_k + \Delta$, set $x_{k+1} = y_j + \lambda_j(b - y_j)$, $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 3.

Step 2d. Set $y_{j+1} = y_j + \lambda_j(b - y_j)$, $j = j + 1$, and go to Step 2a.

Step 3. Compute $f'(x_k)$.

If $f'(x_k) < -\epsilon_3$, go to Step 4.
If $f'(x_k) > \epsilon_3$, go to Step 5.
If $-\epsilon_3 \leqslant f'(x_k) \leqslant \epsilon_3$, go to Step 6.

Step 4. Set $i = 1$, $u_i = x_k$, and go to Step 4a.

Step 4a. Compute $g'(u_i)$ and go to Step 4b.

Step 4b. Solve $L_g(u_i)$ and let u_{i+1} be ϵ_2-optimal (e.g., u_{i+1} is optimal or $\epsilon_2 \geqslant h'(u_{i+1}) + g'(u_i) \geqslant -\epsilon_2$).

If $u_{i+1} = b$, set $x_{k+1} = u_{i+1}$, $UB_{k+1} = \min\{UB_k, f(x_{k+1})\}$, $k = k + 1$ and go to Step 7.

If $u_{i+1} < b$ and $f'(u_{i+1}) < -\epsilon_3$, set $i = i + 1$ and go to Step 4a.

If $u_{i+1} < b$ and $\epsilon_2 \geqslant f'(u_{i+1}) \geqslant -\epsilon_3$, set $x_{k+1} = u_{i+1}$, $UB_{k+1} = \min\{UB_k, f(x_{k+1})\}$, $k = k + 1$ and go to Step 6.

Step 5. Set $r = 1$, $v_r = x_k$, and go to Step 5a.

Step 5a. Compute $h'(v_r)$. If $R_r(b) \geqslant 0$, go to Step 7. If $R_r(b) < 0$, go to Step 5b.

Step 5b. Solve $L_h(v_r)$ and let v_{r+1} be ϵ_4-optimal (e.g., $v_{r+1} = b$ is optimal or $\epsilon_4 \leqslant R_r(v_{r+1}) \leqslant 0$).

If $v_{r+1} = b$, go to Step 7.
If $v_{r+1} < b$, compute

$$\delta(v_{r+1}) = \frac{-f(v_{r+1}) + UB_k}{(b - v_{r+1})\,h'(v_{r+1}) + g(b) - g(v_{r+1})}.$$

If $\delta(v_{r+1}) \geqslant \delta_1$, set $x_{k+1} = v_{r+1}$, $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 2.

If $\delta(v_{r+1}) < \delta_1$ and $f'(v_{r+1}) > \epsilon_3$, set $r = r + 1$ and go to Step 5a.

If $\delta(v_{r+1}) < \delta_1$ and $-\epsilon_3 \leqslant f'(v_{r+1}) \leqslant \epsilon_3$, set $x_{k+1} = v_{r+1}$, $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 6.

If $\delta(v_{r+1}) < \delta_1$ and $f'(v_{r+1}) < -\epsilon_3$, set $x_{k+1} = v_{r+1}$, $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 4.

Step 6. Set $x_{k+1} = \min\{x_k + \delta, b\}$ and go to Step 6a.

Step 6a.

If $x_{k+1} = b$ and $f(x_{k+1}) \leqslant UB_k$, set $UB_{k+1} = f(x_{k+1})$, $k = k + 1$ and go to Step 7.

If $x_{k+1} = b$ and $f(x_{k+1}) > UB_k$, go to Step 7.

If $x_{k+1} < b$ and $f(x_{k+1}) < f(x_k)$, set $UB_{k+1} = \min\{UB_k, f(x_{k+1})\}$, $k = k + 1$, and go to Step 1.

If $x_{k+1} < b$ and $f(x_{k+1}) > f(x_k)$, set $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 1.

If $x_{k+1} < b$ and $f(x_{k+1}) = f(x_k)$, set $UB_{k+1} = UB_k$, $k = k + 1$, and go to Step 6.

Step 7. Let $x^* \in \{x \in [a, b] \mid f(x) \leqslant UB_k\}$. Then x^* is ϵ-optimal; stop.

Acknowledgment

The authors are grateful to Mr. Lee-Ping Chyan for coding the algorithm.

References

1. M. Aoki, "Introduction to Optimization Techniques: Fundamentals and Applications of Nonlinear Programming," Macmillan, New York, 1971.
2. D. J. Wilde, "Optimum Seeking Methods," Prentice–Hall, N.J., 1964.
3. B. O. Shubert, A sequential method seeking the global maximum of a function, *SIAM J. Numer. Anal.* 9 (1972), 379–388.
4. R. P. Brent, "Algorithm for Minimization Without Derivatives," Prentice-Hall, N.J., 1973.