M

Office of Naval Research

Contract Nonr-1866(16) NR-372-012

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Grant NGR 22-007-068

FUNCTION MINIMIZATION WITHOUT DERIVATIVES BY A SEQUENCE OF QUADRATIC PROGRAMMING PROBLEMS

By

David H. Winfield

August 1967

Technical Report No. 537

Division of Engineering and Applied Physics
Harvard University • Cambridge, Massachusetts

Office of Naval Research

Contract Nonr-1866(16)

NR - 372 - 012

National Aeronautics and Space Administration

Grant NGR-22-007-068

FUNCTION MINIMIZATION WITHOUT DERIVATIVES BY A SEQUENCE OF QUADRATIC PROGRAMMING PROBLEMS

By

David H. Winfield

Technical Report No. 537

August 1967

The research reported in this document was made possible through support extended the Division of Engineering and Applied Physics, Harvard University by the U. S. Army Research Office, the U. S. Air Force Office of Scientific Research and the U. S. Office of Naval Research under the Joint Services Electronics Program by Contracts Nonr-1866(16), (07), and (32) and by the National Aeronautics and Space Administration under Grant NGR-22-007-068.

Division of Engineering and Applied Physics

Harvard University Cambridge, Massachusetts

FUNCTION MINIMIZATION WITHOUT DERIVATIVES BY A SEQUENCE OF QUADRATIC PROGRAMMING PROBLEMS

By

David H. Winfield

Division of Engineering and Applied Physics
Harvard University Cambridge, Massachusetts

ABSTRACT

An algorithm is described for minimizing an arbitrary scalar cost function c(x) with respect to an n-vector x. At each stage of the minimization, the cost function is approximated by a quadratic form in the region about the current lowest-cost point. The next trial point is taken as the minimum of this quadratic form within a hypercube in n-space centered at the current lowest-cost point.

The procedure has quadratic convergence, but differs from other quadratically convergent minimization algorithms in that (1) minimization is over a sequence of n-dimensional regions rather than over a sequence of one-dimensional straight lines (2) the local approximation to the cost surface need not be positive definite (3) each approximation depends only on true cost values and is independent of prior approximations (4) after each evaluation of cost at a trial point, the trial point is added, and a point distant from the current lowest-cost point is deleted, from the set of points to which the next quadratic form will interpolate. The algorithm takes relatively large steps, and is forced by (4) to learn from its failures.

Test results are presented for n = 2 using Rosenbrock's parabolic valley as the cost function.

Introduction

Recent surveys of function minimization algorithms [1], [2], show the superiority of algorithms having the property of quadratic convergence, i. e., the property that if the cost function is exactly a quadratic form, the computation terminates at the exact minimum in a finite number of steps.* These algorithms involve a sequence of one-dimensional searches along a sequence of straight lines. In the last cycle of one-dimensional search, the directions of search are mutually conjugate. This last cycle of search along conjugate directions yields quadratic convergence [3].

In any search procedure, values of cost and possibly gradients of cost are computed at a sequence of points. Each datum consisting of a value of cost and associated x, or a component of the gradient and associated x, furnishes information that could be used as an equation of condition for the coefficients of a local quadratic model in n-space. As soon as the number of data gathered on the cost function equals the number of coefficients required in the local quadratic model, it is usually possible to compute these coefficients and to find the minimum of the quadratic model within some region in which the model is assumed to be valid.

If the cost function is a positive definite quadratic form, it will be modeled exactly. The minimum of the quadratic model will be the minimum of the cost function. Thus any algorithm which directly computes and minimizes a local quadratic model has quadratic convergence.

The number of data required to define the local quadratic form is the minimum number of data required to give an algorithm the property of

* See Reference [5], pp. 72-73 for the distinction between quadratic convergence in this sense, and the asymptotic quadratic behavior of an iteration such as Newton's method for finding the root of an equation.

quadratic convergence. All the algorithms based on a sequence of one-dimensional searches use more than this minimum number of data. This suggests the possibility that in minimizing a general (i. e., non-quadratic) cost function, an algorithm based on local quadratic models may require less data on the cost function than algorithms based on one-dimensional searches.

As a test of this conjecture, local quadratic models are used in the problem of function minimization without derivatives, using Rosenbrock's valley [4] as the cost function. At each stage, a local quadratic model is minimized over a square constraint region, giving rise to a Sequence of Quadratic Programming problems. (Hence the name of the algorithm to be described is SQP.)

Description of the SQP Algorithm for n = 2

In the local quadratic model

$$q(x) = \tfrac{1}{2} x^T Ax + b^T x + d \qquad (1)$$

there are $\frac{1}{2}n(n+1)$ distinct elements in the symmetric matrix A, n elements in the vector b, and one element in the scalar d, or a total of

$$N = \tfrac{1}{2}(n+1)(n+2) \qquad (2)$$

unknown coefficients to be determined by fitting q(x) to c(x) at a set of points x^i, $i = 1, \ldots, N$. In a plane, n = 2 and N = 6.

The algorithm proceeds as follows:

1. Initialize the search by evaluating cost at the starting point x and at five additional points obtained by incrementing the starting point by vectors (-a, 0), (a, 0), (0, -a), (0, a) and (a, a).

2. Define the basepoint as the lowest-cost point of these six points. Store the coordinates of the basepoint and cost at the basepoint into the first row of a table of points and costs. In the second row, store the point

nearest the basepoint and cost at that point. Continue storing points and costs ordered on distance from the basepoint in successive lower rows. After a translation of coordinates which places the origin at the basepoint, the table has the form

0	0	$c(0)$
x_1^2	x_2^2	$c(x^2)$
x_1^3	x_2^3	$c(x^3)$
$\vdots$	$\vdots$	$\vdots$
x_1^6	x_2^6	$c(x^6)$
$\vdots$	$\vdots$	$\vdots$
x_1^{10}	x_1^{10}	$c(x^{10})$

Figure 1. Table of Points and Costs

The entries satisfy

$$\begin{array}{ll} c(0) \leqq c(x^i) & i = 2, \ldots, 10 \\ [x^i]^T x^i \leqq [x^j]^T x^j & 1 \leqq i < j \leqq 10 \end{array} \qquad (3)$$

This table is arbitrarily limited to ten rows, and is reordered to satisfy the above relations whenever a new basepoint is chosen. The table is initialized with entries in the first six rows, and acquires entries in the lower rows as the search progresses.

Only the first six rows participate in forming the local quadratic model. The remaining four rows constitute a residual memory which may be called upon in the event that a basepoint is found near one of these four points.

It will be shown that each new point first appears in one of the six top rows, gradually goes lower in the table as the basepoint moves away from it, and is finally discarded.

3. With the origin at the basepoint, d in equation (1) is simply c(0). To find a_{11}, a_{12}, a_{22}, b_1, b_2, for the quadratic model, solve the five equations of condition

$$\frac{1}{2}[x^i]^T A x^i + b^T x^i = c(x^i) - d \qquad i = 2, 3, \ldots, 6 \qquad (4)$$

where x^i and $c(x^i)$ are the points and costs in the second through sixth rows of the table of points and costs. The system (4) is non-singular for the initial points specified in step (1.) due to the pattern of these points. When the system (4) was solved with other sets of five points generated in the course of the Rosenbrock problem, no singularity was ever encountered.

4. As a check on the A, b, d just computed, evaluate the quadratic model q(x) to verify that it equals c(x) at the points of interpolation. In the Rosenbrock problem, this condition was always satisfied to six digits.

In the following computations, the region of validity of the quadratic model will be defined as a certain square centered at the current basepoint. The minimum of the quadratic form over this square constraint region will be found by solving a quadratic programming problem.

Let x_{min} be the solution of the quadratic programming problem. If the true cost $c(x_{min})$ is less at x_{min} than at any previous trial point, then x_{min} becomes the new basepoint and is placed in the first row of the table of points and costs. If $c(x_{min})$ is not less than cost at the current basepoint, then (as will be shown) the ordering of points by distance from the current basepoint causes x_{min} and $c(x_{min})$ to be entered in the table in one of the rows 2, 3, 4, 5, or 6. Since the first six rows determine the next quadratic model, the recent trial x_{min} always influences the next model.

Regardless of whether or not $c(x_{min})$ represents a reduction of cost, a new quadratic model, a new constraint region, and hence a new quadratic programming problem, are defined. Thus each successive trial point (after the initial six) is obtained by solving a quadratic programming problem. The algorithm as a whole is a sequence of quadratic programming problems.

5. Let r be the distance* from the basepoint to the farthest of the five points nearest the basepoint, i. e.,

$$r = [x^6]^T x^6 \qquad (5)$$

Let α be a constraint region reduction factor. Set $\alpha = 1$ for the first quadratic programming problem. Within the sequence of quadratic programming problems, set $\alpha = 1$ if the previous quadratic programming problem has located a new lowest-cost point. If the previous quadratic programming problem failed to locate a new lowest-cost point, set

$$\alpha_{new} = .95\alpha_{old} \qquad (6)$$

6. Define a square constraint region centered at the basepoint, with corners at $(\pm G, \pm G)$ relative to the basepoint $(0, 0)$, where the semi-side G is computed as

$$G = \frac{.999\, r\alpha}{\sqrt{2}} \qquad (7)$$

Then limit G by $G \leqq G_{max}$, where G_{max} is an appropriate bound for a particular problem.

7. Minimize the local quadratic model over the square constraint region. This is a well-posed quadratic programming problem regardless of whether or not the quadratic form is positive definite. The minimum

* Distance may be measured relative to an arbitrary metric, here taken as the identity matrix.

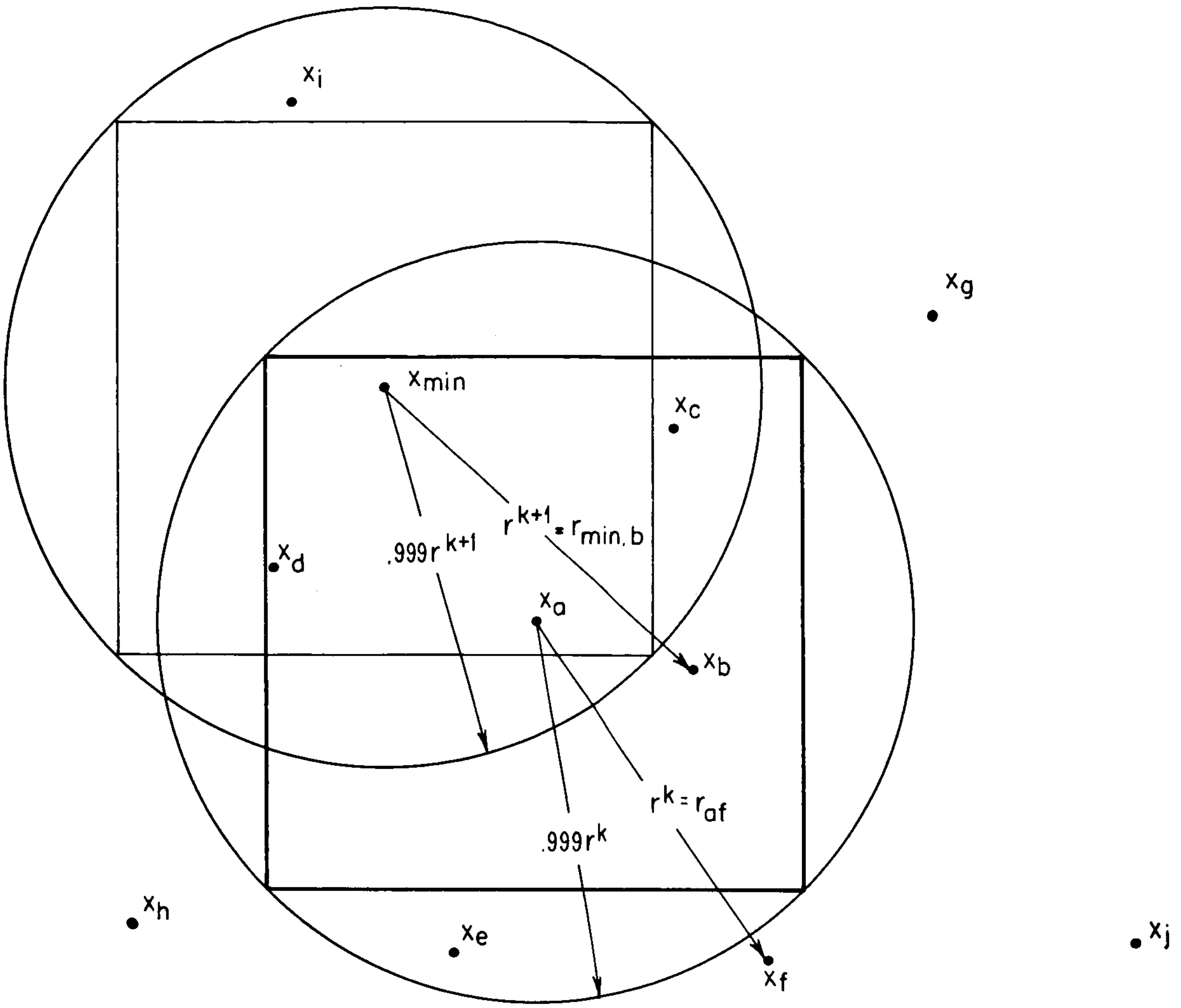
x_{min} may lie on a boundary or in the interior of the square. As convergence is approached, the minimum will occur in the interior of the square.

8. Evaluate $c(x_{min})$. If this is less than cost at the current basepoint, then x_{min} becomes the new basepoint. In the table of points and costs, x_{min} and $c(x_{min})$ are placed in the first row, and other points are reordered according to their distances from the new basepoint.

Figure 2 illustrates the events corresponding to a reduction of cost. First the basepoint $x_a = \bar{0}$ and adjacent points x_b, x_c, x_d, x_e, x_f,* and their associated cost values participate in forming a local quadratic model. Suppose $\alpha = 1$ at this k-th stage. The distance from the basepoint to x_f is r^k. The constraint region is a square with sides parallel to the coordinate axes, inscribed in a circle of radius $.999r^k$ centered at the basepoint. The minimum of the quadratic form over the constraint region is found at x_{min}, and $c(x_{min})$ proves to be less than cost at all previous trial points. Now points are reordered by distance relative to the new basepoint x_{min}. Points x_{min}, x_d, x_a, x_c, x_i, x_b, participate in forming a new quadratic model. The new r^{k+1} is the distance from x_{min} to x_b. Since the previous quadratic programming problem reduced cost, $\alpha = 1$, and the new constraint region is the square inscribed in the circle of radius $.999r^{k+1}$ centered at x_{min}.

9. Suppose that x_{min}, the minimum of the quadratic model $q(x)$, proved not to be a minimum of the true cost function, $c(x)$. Then x_a is retained as basepoint, as shown in Figure 3. A new local quadratic model is based on x_a, x_b, x_c, x_d, x_{min}, x_e. The new r^{k+1} is the distance from x_a to x_e. Because the previous trial did not reduce cost, $\alpha = .95$. The new

* The letter subscripts denote fixed points regardless of distance from the basepoint, in contrast to the numerical superscripts of Figure 1, which denote ordering by distance from the current basepoint.



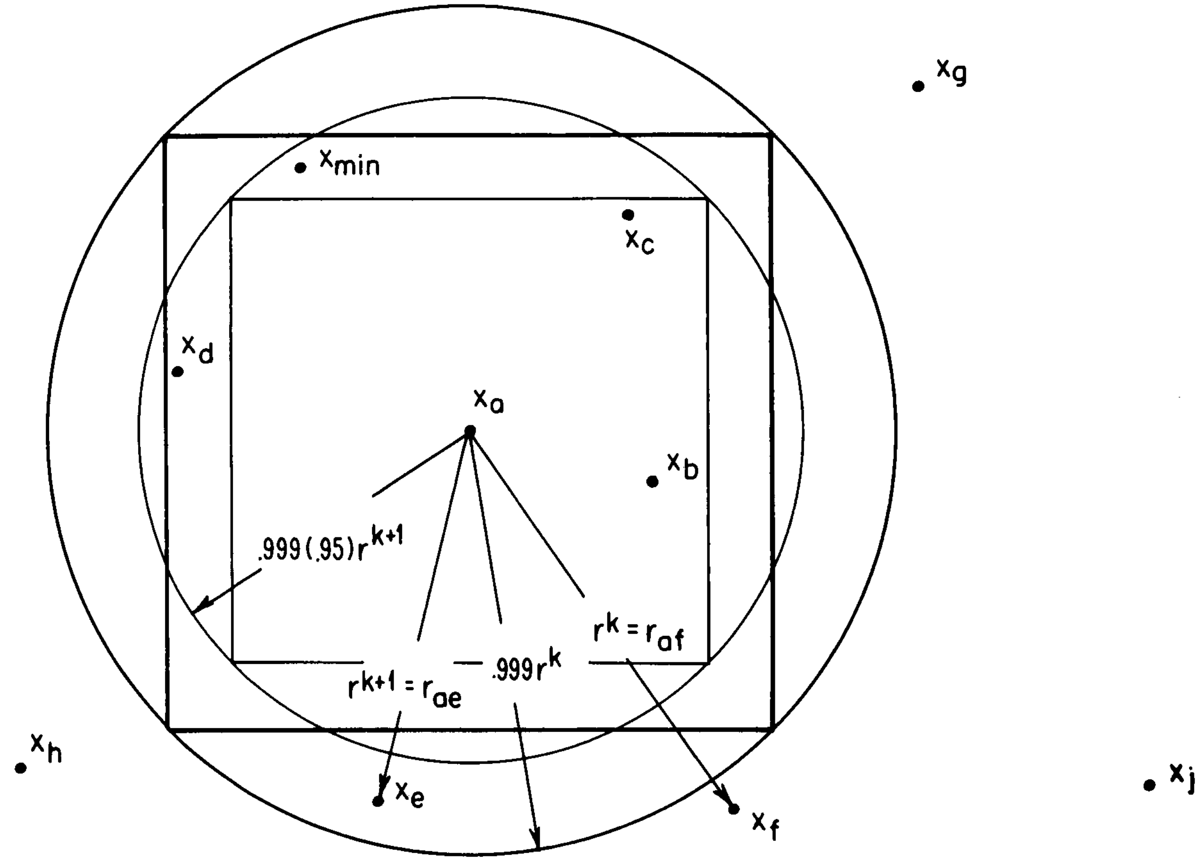
K- th Table of Points and Costs

x_a	$c(x_a)$	Influence local quadratic model
x_b	$c(x_b)$	
x_c	$c(x_c)$	
x_d	$c(x_d)$	
x_e	$c(x_e)$	
x_f	$c(x_f)$	
x_g	$c(x_g)$	
x_h	$c(x_h)$	
x_i	$c(x_i)$	
x_j	$c(x_j)$	

k+1 st Table of Points and Costs

x_{min}	$c(x_{min})$	Influence local quadratic model
x_d	$c(x_d)$	
x_a	$c(x_a)$	
x_c	$c(x_c)$	
x_i	$c(x_i)$	
x_b	$c(x_b)$	
x_g	$c(x_g)$	
x_e	$c(x_e)$	
x_h	$c(x_h)$	
x_f	$c(x_f)$	

FIG. 2 THE NEW TRIAL POINT x_{min} REDUCES COST



K-th Table of Points and Costs		
x_a	$c(x_a)$	Influence local quadratic model
x_b	$c(x_b)$	
x_c	$c(x_c)$	
x_d	$c(x_d)$	
x_e	$c(x_e)$	
x_f	$c(x_f)$	
x_g	$c(x_g)$	
x_h	$c(x_h)$	
x_i	$c(x_i)$	
x_j	$c(x_j)$	

k+1 st Table of Points and Costs		
x_a	$c(x_a)$	Influence local quadratic model
x_b	$c(x_b)$	
x_c	$c(x_c)$	
x_d	$c(x_d)$	
x_{min}	$c(x_{min})$	
x_e	$c(x_e)$	
x_f	$c(x_f)$	
x_g	$c(x_g)$	
x_h	$c(x_h)$	
x_i	$c(x_i)$	

FIG. 3 THE NEW TRIAL POINT x_{min} FAILS TO REDUCE COST

constraint region is the square inscribed in the circle of radius $(.999)(.95)r^{k+1}$ centered at x_a.

10. The shape of the constraint region was chosen to be a square because this simplifies the quadratic programming problem. The particular formula (7) for the size of the square was chosen to force the algorithm to "learn" quickly from failures.

Suppose the constraint region had been permitted to be so large that its corner was farther from the basepoint than the farthest of the five closest points used to form the interpolating quadratic form. Suppose the minimum of the quadratic form occurred at a corner, and that this point failed to reduce cost. Then the next cycle would begin with the same basepoint, with the same five points closest to the basepoint, and hence with the same quadratic form. The constraint region would have contracted, but the quadratic form would be unchanged. After several (wasted) cycles the constraint region would have shrunk sufficiently to force inclusion of a new point as one of the closest five. In contrast, formula (7) forces immediate inclusion of the failure point in determining the next quadratic form. Formula (7) ensures that every point in the constraint region, and hence x_{min}, is closer to the basepoint than the farthest of the five points used in defining the quadratic form. Thus if the new trial point fails to reduce cost, and the original basepoint is retained, the new trial point is added, and the most distant of the original five points is deleted, from the set of points used to form the next quadratic model.

Numerical Comparison of SQP with Other Function Minimization Methods Which Do Not Require Derivatives

A frequently used cost function for testing minimization algorithms is the parabolic valley of Rosenbrock [4].

$$c(x) = 100(x_1 - x_2^2)^2 + (1 - x_1)^2 \tag{8}$$

For this cost function, Powell [3] and Stewart [5] report data which are reproduced in Tables 1 and 2. Table 3 is a computer printout of data obtained using SQP. The number of function evaluations required by SQP to reach the minimum is about .38 times the number required by Stewart's method. The computer time required on the IBM 7094 for setting up and solving the 62 quadratic programming problems was about .8 ± .1 minute.

Application in Higher-Dimensional Minimization Problems

A virtue of SQP is its efficient use of each datum of information concerning the cost function, so that the overall minimization may be completed with relatively few data. A defect is the computational effort required to fit the quadratic model.

In n-space, the number of data in addition to cost at the basepoint required to fit a local quadratic model are

$$N - 1 = \tfrac{1}{2}(n + 1)(n + 2) - 1 = \tfrac{1}{2}n(n + 3)$$

The $\frac{1}{2}n(n + 3)$ coefficients of the quadratic model are obtained by solving $\frac{1}{2}n(n + 3)$ linear equations. The computational work in solving a linear system varies as the cube of the dimension [6]. Hence the computational work per quadratically convergent cycle varies as the sixth power of n.

The dollar cost of function minimization by SQP is approximately

$$D = N_q\{\tfrac{1}{2}(n + 1)(n + 2)k_1 + [\tfrac{1}{2}n(n + 3)]^3k_2 + (n - 1)^3k_3\}$$

Iteration	Function Values	x_1	x_2	$f(x_1, x_2)$
0	1	-1·2000	1·0000	24·2000
1	14	-0·9912	0·9927	3·9753
2	25	-0·7674	0·5485	3·2863
3	35	-0·5017	0·2064	2·4608
4	46	-0·2840	0·0307	1·8978
5	57	-0·0123	-0·0408	1·1927
6	71	0·2568	0·0369	0·6366
7	84	0·4379	0·1624	0·4026
8	97	0·6810	0·4478	0·1274
9	109	0·8341	0·6818	0·0469
10	122	0·8894	0·7948	0·0137
11	131	1·0014	0·9997	0·0010
12	142	0·9926	0·9850	6×10^{-5}
13	151	1·0000	1·0000	7×10^{-10}

Table 1. Powell's Descent of the Rosenbrock Valley [3]

Iteration No.	Number of function evaluations	Φ
0	1	2.4 10^{1}
2	13	3.8 10^{0}
4	26	2.9 10^{0}
6	39	1.9 10^{0}
8	56	7.1 10^{-1}
10	70	2.9 10^{-1}
12	83	1.4 10^{-1}
14	97	5.4 10^{-2}
16	111	1.8 10^{-2}
18	124	1.3 10^{-3}
20	132	1.7 10^{-6}
22	145	2.8 10^{-10}
23	152	1.0 10^{-11}
24	163	9.0 10^{-12}
25	169	3.3 10^{-12}
26	174	3.3 10^{-12}

Table 2. Stewart's Descent of the Rosenbrock Valley [5]

Run Summary Table

Evaluation Number	X	Y	Cost
1	-0.1700000E 01	0.1000000E 01	0.3645000E 03
2	-0.1200000E 01	0.1000000E 01	0.2420000E 02
3	-0.7000000E 00	0.1000000E 01	0.2890000E 02
4	-0.1200000E 01	0.1500000E 01	0.5200000E 01
5	-0.1200000E 01	0.5000000E 00	0.9320000E 02
6	-0.1200000E 01	0.1500000E 01	0.5200000E 01
7	-0.9568116E 00	0.1000000E 01	0.4543332E 01
8	-0.7486519E 00	0.6029354E 00	0.3238033E 01
9	-0.7486519E 00	0.6029354E 00	0.3238033E 01
10	-0.6719982E 00	0.4597000E 00	0.2802169E 01
11	-0.6719982E 00	0.4597000E 00	0.2802169E 01
12	-0.6719982E 00	0.4597000E 00	0.2802169E 01
13	-0.6382675E 00	0.4055506E 00	0.2684257E 01
14	-0.6382675E 00	0.4055506E 00	0.2684257E 01
15	-0.6382675E 00	0.4055506E 00	0.2684257E 01
16	-0.4309351E 00	0.1982181E 00	0.2063233E 01
17	-0.4309351E 00	0.1982181E 00	0.2063233E 01
18	-0.4309351E 00	0.1982181E 00	0.2063233E 01
19	-0.3445203E 00	0.1652721E 00	0.2024684E 01
20	-0.3445203E 00	0.1652721E 00	0.2024684E 01
21	-0.1235784E 00	-0.1759378E-01	0.1370442E 01
22	-0.1235784E 00	-0.1759378E-01	0.1370442E 01
23	-0.1235784E 00	-0.1759378E-01	0.1370442E 01
24	-0.1235784E 00	-0.1759378E-01	0.1370442E 01
25	0.5012290E-01	0.1311605E-01	0.9135104E 00
26	0.5012290E-01	0.1311605E-01	0.9135104E 00
27	0.2273967E 00	0.9037245E-01	0.7463998E 00
28	0.2273967E 00	0.9037245E-01	0.7463998E 00
29	0.2273967E 00	0.9037245E-01	0.7463998E 00
30	0.2273967E 00	0.9037245E-01	0.7463998E 00
31	0.2273967E 00	0.9037245E-01	0.7463998E 00
32	0.3931369E 00	0.1488131E 00	0.3715816E 00
33	0.5699951E 00	0.2854609E 00	0.3404050E 00
34	0.5699951E 00	0.2854609E 00	0.3404050E 00
35	0.4681424E 00	0.2230931E 00	0.2844215E 00
36	0.4855394E 00	0.2363176E 00	0.2647021E 00
37	0.6000640E 00	0.3523353E 00	0.1659418E 00
38	0.6000640E 00	0.3523353E 00	0.1659418E 00
39	0.6000640E 00	0.3523353E 00	0.1659418E 00
40	0.6697577E 00	0.4700739E 00	0.1552787E 00
41	0.6397044E 00	0.4172360E 00	0.1362358E 00

Table 3. SQP Descent of the Rosenbrock Valley

Evaluation Number	X	Y	Cost
42	0.6397044E 00	0.4172360E 00	0.1362358E 00
43	0.7420811E 00	0.5475165E 00	0.6752569E-01
44	0.7420811E 00	0.5475165E 00	0.6752569E-01
45	0.7420811E 00	0.5475165E 00	0.6752569E-01
46	0.7723481E 00	0.5873935E 00	0.6015758E-01
47	0.7723481E 00	0.5873935E 00	0.6015758E-01
48	0.8048254E 00	0.6460256E 00	0.3838839E-01
49	0.8191585E 00	0.6699046E 00	0.3282820E-01
50	0.8690845E 00	0.7512339E 00	0.1879859E-01
51	0.8690845E 00	0.7512339E 00	0.1879859E-01
52	0.8690845E 00	0.7512339E 00	0.1879859E-01
53	0.9183851E 00	0.8377926E 00	0.9840279E-02
54	0.9623699E 00	0.9236246E 00	0.2056763E-02
55	0.9623699E 00	0.9236246E 00	0.2056763E-02
56	0.9964662E 00	0.9914445E 00	0.2375762E-03
57	0.9964662E 00	0.9914445E 00	0.2375762E-03
58	0.9964662E 00	0.9914445E 00	0.2375762E-03
59	0.9960381E 00	0.9918040E 00	0.2398718E-04
60	0.9960381E 00	0.9918040E 00	0.2398718E-04
61	0.9984150E 00	0.9968594E 00	0.2584063E-05
62	0.9991590E 00	0.9983379E 00	0.7439477E-06
63	0.1000015E 01	0.1000005E 01	0.6274457E-07
64	0.1000015E 01	0.1000005E 01	0.6274457E-07
65	0.9999919E 00	0.9999819E 00	0.4103697E-09
66	0.9999960E 00	0.9999907E 00	0.1820280E-09
67	0.1000001E 01	0.1000002E 01	0.1982636E-11
68	0.9999999E 00	0.9999999E 00	0.4496403E-14

Table 3. (continued)

where N_q is the number of quadratic programming problems required for convergence.

Within the curly brackets, the first term represents the cost of obtaining $\frac{1}{2}(n + 1)(n + 2)$ data on the function. The second term represents the cost of fitting the quadratic model. The third term represents the cost of solving the quadratic programming problem. For large n, the second term is dominant.

The SQP technique seems appropriate for problems in which the cost of evaluating the function to be minimized (e. g. by actually carrying out some experimental process) is large relative to the cost of computation to locate a new trial point.

Conclusion

The distinguishing features of the SQP algorithm are

(1) generation of a table of true cost values at a sequence of points

(2) generation of an interpolating function which approximates the true cost function within a region (e. g. , a hypercube) centered at the current lowest-cost point

(3) selection of a new trial point as the minimum of the interpolating function within the region

(4) a method of forcing the modification of the interpolating function by inclusion of information from each new trial, whether or not that trial reduced cost.

The algorithm differs from current practice in the character of memory and in the method of learning from failure.

Memory consists only of the table of points and costs. At each step, past trials influence the location of the next trial only through the particular $\frac{1}{2}(n + 1)(n + 2)$ points and cost values which determine the current interpolating quadratic form. These points alone determine the local quadratic model and the region in which it is to be minimized. Each local model is formed afresh, based only on "raw data" from the table of points and costs. Only points close to the current basepoint may determine the quadratic model to be used in finding a new basepoint. This is in contrast to the method of Davidon [7], [8] in which the current approximation to the matrix of second partial derivatives of cost with respect to x reflects all prior experience, even obsolete experience at an initial point far distant from the current trial. This is also in contrast to Powell's algorithm [3], in which the current set of (approximately) conjugate directions reflect all prior experience in the search.

In the algorithms of Davidon and Powell, cost values are not retained after their immediate use in one-dimensional minimization. They are not used to guide the location of future trials. In contrast, in SQP, the size of the constraint hypercube forces the new trial point to be sufficiently close to the basepoint so that, even if the trial fails to reduce cost, the new point will be included as one of the $\frac{1}{2}n(n + 3)$ points closest to the basepoint, and will play a role in determining the next interpolating quadratic form.[*] So long as the basepoint remains sufficiently near this point, the point continues to influence the interpolating quadratic form.

The influence history of a given trial point x and cost c(x) may be summarized as follows: As the basepoint moves away from x (as the result of successful trials) or more points are added nearer to the basepoint than x (as the result of unsuccessful trials) the trial point x is eventually excluded from the set of points determining the current quadratic model. The point remains in reserve in the table of points and costs, ready to be

[*] This forced revision of the consensus reached by the previous set of $\frac{1}{2}(n + 1)(n + 2)$ points is an important and possibly novel feature of SQP.

consulted should a basepoint ever again come sufficiently close. Only after the point drops to some sufficiently low status in terms of closeness to the current basepoint, is it forgotten entirely.

Acknowledgments

I wish to thank my thesis advisor, Professor Arthur E. Bryson Jr., and my instructor in numerical analysis, Assistant Professor Donald G. Anderson, for their helpful discussions.

References

1. Fletcher, R. Function minimization without evaluating derivatives -- a review. Computer Journal 8 (1965), 33-41.

2. Box, M. J. A new method of constrained optimization and a comparison with other methods. Computer Journal 8 (1965), 42-52.

3. Powell, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal 7 (1964), 155-162.

4. Wilde, D. J. Optimum Seeking Methods, Prentice-Hall, Englewood Cliffs, N. J., 1964, p. 152.

5. Stewart, G. W. III. A modification of Davidon's minimization method to accept difference approximations of derivatives. Journal of the Association for Computing Machinery, Vol. 14, No. 1, January 1967, 72-83.

6. Fadeeva, V. N. Computational Methods of Linear Algebra, Dover Publishing Co., New York, 1958, p. 70.

7. Davidon, W. C. Variable metric method for minimization. Atomic Energy Commission Research and Development Report, ANL-5990 (Revised), 1959.

8. Fletcher, R. and Powell, M. J. D. A rapidly convergent descent method for minimization. Computer Journal 6 (1963), 163-168.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Division of Engineering and Applied Physics
Harvard University Cambridge, Massachusetts

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
FUNCTION MINIMIZATION WITHOUT DERIVATIVES BY A SEQUENCE OF QUADRATIC PROGRAMMING PROBLEMS

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*
Interim technical report

5. AUTHOR(S) *(First name, middle initial, last name)*
David H. Winfield

6. REPORT DATE
August 1967

7a. TOTAL NO. OF PAGES
21

7b. NO. OF REFS
8

8a. CONTRACT OR GRANT NO.
Nonr-1866(16) & NASA NGR-22-007-068

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Technical Report No. 537

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Reproduction in whole or in part is permitted by the U. S. Government. Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research

13. ABSTRACT

An algorithm is described for minimizing an arbitrary scalar cost function c(x) with respect to an n-vector x. At each stage of the minimization, the cost function is approximated by a quadratic form in the region about the current lowest-cost point. The next trial point is taken as the minimum of this quadratic form within a hypercube in n-space centered at the current lowest-cost point.

The procedure has quadratic convergence, but differs from other quadratically convergent minimization algorithms in that (1) minimization is over a sequence of n-dimensional regions rather than over a sequence of one-dimensional straight lines (2) the local approximation to the cost surface need not be positive definite (3) each approximation depends only on true cost values and is independent of prior approximations (4) after each evaluation of cost at a trial point, the trial point is added, and a point distant from the current lowest-cost point is deleted, from the set of points to which the next quadratic form will interpolate. The algorithm takes relatively large steps, and is forced by (4) to learn from its failures.

Test results are presented for n = 2 using Rosenbrock's parabolic valley as the cost function.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Function minimization						
Nonlinear programming						
Mathematical programming						
Process control						
Parameter search						
Optimization						