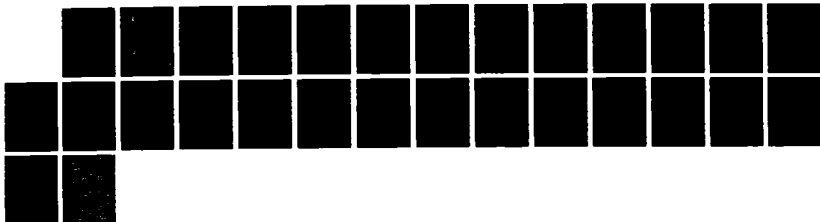
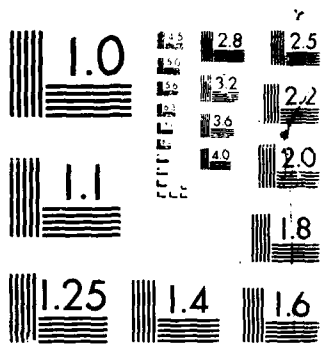


AD-A195 184

SINMIN - A PROGRAM FOR MINIMIZING FUNCTIONS BY SIMPLEX 171
METHODS FOR IBM PC. (U) ARMY MISSILE COMMAND REDSTONE
ARSENAL AL DIRECTED ENERGY DIRE. J D SMITH ET AL
09 JUL 87 ANSHI/TR-RD-DE-87-4 F/G 12/4 ML

UNCLASSIFIED





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

2

TECHNICAL REPORT RD-DE-87-4

DTIC FILE CODE

SIMMIN - A PROGRAM FOR MINIMIZING
FUNCTIONS BY SIMPLEX METHODS FOR
IBM PC

Jerry D. Smith
Miles E. Holloman
William F. Otto
Directed Energy Directorate
Research, Development,
and Engineering Center

AD-A195 184

JULY 1987



U.S. ARMY MISSILE COMMAND

Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.

DTIC
ELECTE
MAY 06 1988
S H D

DISPOSITION INSTRUCTIONS

**DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT
RETURN IT TO THE ORIGINATOR.**

DISCLAIMER

**THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN
OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED
BY OTHER AUTHORIZED DOCUMENTS.**

TRADE NAMES

**USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES
NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF
THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.**

REPORT DOCUMENTATION PAGE

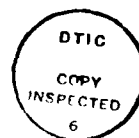
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-RD-DE-87-4		7a. NAME OF MONITORING ORGANIZATION	
6a. NAME OF PERFORMING ORGANIZATION Directed Energy Res, Dev, and Eng Ctr	6b. OFFICE SYMBOL (if applicable) AMSMI-RD-DE	7b. ADDRESS (City, State, and ZIP Code)	
6c. ADDRESS (City, State, and ZIP Code) Commander U.S. Army Missile Command, ATTN: AMSMI-RD-DE Redstone Arsenal, AL 35898		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)	10. SOURCE OF FUNDING NUMBERS	
8c. ADDRESS (City, State, and ZIP Code)		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) SIMMIN - A Program for Minimizing Functions by Simplex Methods for IBM PC			
12. PERSONAL AUTHOR(S) Jerry D. Smith, Miles E. Holloman, William F. Otto			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1987/07/09	15. PAGE COUNT 27
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Minicomputer, Simplex, Nonlinear programming, Optimal solutions, Optimization methods	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) SIMMIN is a minicomputer program capable of finding a minimum for any function which can be described using Fortran. A simplex method determines values of coefficients which result in a function local minimum. This document briefly describes the methodology and algorithm originally written by Caceci and Cacheris and modified by Holloman and Otto. It also provides a user's guide for use of the Fortran code (which is included), complete with example cases. Conversion to larger computers is straightforward.			
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Jerry D. Smith		22b. TELEPHONE (Include Area Code) (205) 876-4329	22c. OFFICE SYMBOL AMSMI-RD-DE-SD

SUMMARY

A computer program, SIMMIN, which is capable of finding a minimum (if one exists) for any function which can be described using Fortran, has been developed and tested on an IBM PC with an 8087 math coprocessor. The program is a modification of a Pascal algorithm written by Caceci and Cacheris and modified by Holloman and Otto. A simplex method determines values of initially unknown coefficients which result in a local function minimum.

The user can vary initial estimates of the coefficients, step sizes, maximum number of iterations, and the convergence criteria, of which two are available. Final values of the coefficients, an estimate of the function minimum, and the number of iterations required are displayed on screen, while a diagnostic iteration history is recorded on a separate file which can be examined after each execution.

The program can easily be converted for use on larger computers, perhaps allowing the use of single precision. Although divergence is impossible, a price is often paid in more iterations and computing time than with some other traditional minimization methods.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION.....	1
II. THE MODEL.....	1
III. COMPUTER IMPLEMENTATION.....	2
A. Input Parameters.....	3
B. Test Results.....	4
IV. CONCLUSIONS.....	4
REFERENCES.....	6
APPENDIX.....	A-1

I. INTRODUCTION

Techniques for minimizing functions include stepwise descent, steepest descent, Newton-Raphson algorithms (and variations), and Marquardt algorithms. However, there are many functions for which these well-known methods are difficult to implement, are computer intensive, and are prone to diverge. Such functions may be nonlinear, partially empirically defined, not continuous in regions of interest, or quite complicated. Inaccurate calculation of derivatives and the lack of good initial estimates for the unknown coefficients are often responsible for the failure of these methods.

The program SIMMIN, the subject of this report, can find a minimum (if one exists) for any function that can be described by the user in Fortran. SIMMIN does this by using a simplex technique to calculate values of initially unknown coefficients which result in a functional minimum. The user should have some familiarity with Fortran so that the subroutine FUNC in the source code shown in the Appendix can be modified for the function of interest. A very simple function has been included for instructional purposes.

II. THE MODEL

The simplex methodology for finding functional minima is used in SIMMIN. The original algorithm was written by Nelder and Mead in 1965 [1] and was implemented in Extended Mercury Autocode on an Orion computer. The starting point for the present study is a Pascal algorithm by Caceci and Cacheris to fit curves to data [2]. Holloman and Otto modified the model by converting to Fortran, introducing a different convergence criterion, to be discussed later, and hard-coding user inputs, such as initial estimates and step sizes, in order to make the model more transparent to the user. They next incorporated their modified model into user-friendly, curve-fitting programs for IBM PC [3,4]. In addition to familiar Fourier, Lagrange and spline methods, data could be fit to any equation selected by the user. The coefficients in the equation were determined by allowing the simplex technique to minimize the nonlinear function sum of squares of residuals as follows:

$$SSR = \sum_{i=1}^N [Y_{calc}(X_i, A_1, A_2, \dots, A_{mm}) - Y_{exper}(X_i)]^2 \quad (1)$$

where A_1, A_2, \dots, A_{mm} are the unknown coefficients and N is the number of experimental data points, X_i . Least squares fitting to any equation was done without using derivatives or matrix operations. However, it was noted that for most fitting functions, execution times increased greatly with the number of coefficients, above approximately three.

In the present study, the fitting aspects of the model were removed. User control over most inputs was restored since finding a minimum (or minima) usually requires experimentation with initial estimates, step sizes, the maximum number of iterations, and the convergence criteria, of which two are available.

The user is referred to References 1 and 2 for details of the model, but a short introduction is offered here. A simplex is a geometric figure which has

one more vertex than the space in which it is defined has dimensions. For example, the user will find in the Appendix (the source code for program SIMMIN) a function of the following form defined in line numbers 281-284:

$$F = (A-1.)^2 + (B-2.)^2 + (C-3.)^2 \quad (2)$$

The simplex for this function would have four vertices in three-dimensional space. Each vertex is characterized by four values: A,B,C,F. Subroutine SIMPLEX defines the simplex as a two-dimensional array SIMP(I,J), with I being the vertex index and J the dimension index. It should be clear that for our example, values A = 1, B = 2, C = 3 yield the one and only function minimum F = 0.

The principal of the simplex method is to move and reshape the simplex in ways resulting in decreased values of the response function as iterations proceed. Each iteration involves combinations of reflection, expansion, contraction, and shrinkage of the vertices according to logical rules, and then testing the newly calculated function value against the previous value. If the new value is better (meaning lower) than the previous one, then the vertices are accepted and the simplex proceeds in the same general direction. If not, then the vertex changing logic proceeds in an orderly, efficient search or a new direction.

III. COMPUTER IMPLEMENTATION

The present study was done on an IBM PC equipped with an 8087 coprocessor using Ryan-McFarland Fortran (V2.11). This Fortran creates very efficient executable code at the expense of longer compilation times. Double precision was used for all reals by appropriately placed implicit double precision statements. The user can easily remove the double precision if SIMMIN is used on fast computers with large word size, where large numbers of iterations and truncation/round-off errors are not likely to cause problems using single precision.

The primary modification the user will make is to substitute a function of interest for the demonstration function in subroutine FUNC (see the Appendix, Lines 272-287). The program requires that the coefficients enter the subroutine as an array COEF(I), I = 1,...,MM, where MM is the number of coefficients to be determined. The user is then free to convert to a more convenient notation. Perhaps the function to be minimized is available as a function subprogram or a subroutine which would then be called from within subroutine FUNC. The calculated value of the function must be returned from FUNC as COEF(MM+1).

Constraints can be added in the form of Fortran statements directing the program flow to statement 999 in FUNC, often in the form of IF statements restricting the coefficients or function to certain regions. Note what happens then: the function value COEF(MM+1) is returned as the previous value plus a very large number 1.D30. Most calls of subroutine FUNC from within subroutine SIMPLEX are followed by IF tests to determine if the new trial function value RNEXT(MM+1) is less than the previous value, referred to in subroutine simplex as SIMP(MM+i,IH(MM+1)). It certainly will not be if a constraint condition has been exceeded and the logic will guide the simplex away from this region.

An important additional word about constraints: the user may well have them without being aware of them for awhile. The simplex moves about in hyperspace without regard to such problems as attempting to take logarithms of zero or of negative numbers, dividing by zero, or attempting to evaluate functions (such as factorials, gamma functions, etc.) in regions leading to computer underflow or overflow. The user should protect against all such possibilities by adding appropriate statements to direct the program to statement 999, or to do the equivalent action described after statement 999 somewhere else.

A. Input Parameters

Values supplied by the user during interactive operation of SIMMIN are shown within boxes in Figures 1, 2, and 3.

1. MM - The number of coefficients to be calculated, 3 for the example in Equation (2).

2. MAXITR - The maximum number of iterations allowed. This may be only approximate since the program now counts every trial set of vertices as an iteration, regardless of whether the function value increases or decreases. Therefore, the number of iterations shown as part of the results could slightly exceed MAXITR.

3. Initial Estimates of the Coefficients - The user must insure that their order conforms to the definition of coefficients in subroutine FUNC. One should always experiment with these initial estimates to see the effect on results.

4. Convergence Criterion Option

ICON = 1 -- The program compares relative errors of calculated coefficients with a limit entered by the user, typically between 1.D-05 and 1.D-04. If every relative error is less than the limit, then convergence is obtained. Before defining this error, recall for Equation (2) that there are four vertices for the simplex: A, B, C, F, and that each vertex is characterized by four values: A, B, C, F. The simplex is defined as a two-dimensional array SIMP(I,J), with I being the vertex index and J being the dimension index. The error is defined as follows (Appendix, Line 231):

$$\text{ERROR}(I) = (\text{SIMP}(I, \text{IH}(I)) - \text{SIMP}(I, \text{L}(I))) / \text{SIMP}(I, \text{IH}(I)) \quad (3)$$

subroutine ORDER has provided IH(I) and L(I), the indices of the highest and lowest values, respectively, currently associated with vertex I. The value of ERROR(MM+1), the function error, does not seem to decrease to nearly as small as the coefficient errors ERROR(I), I=1,MM even though inspection of latter iterations in file ITER.OUT shows a minimum is close to being found, by either convergence criterion. Therefore, the error comparisons were restricted to the coefficients only.

ICON = 2 -- This convergence option allows the program to keep a count of the number of consecutive times that new trial vertices have been calculated without reducing the value of the function. If this count exceeds (MM+1)*3 then convergence is obtained. It was arrived at through experience

with nonlinear functions in fitting applications, and relieves the user from choosing acceptable limits for errors [3,4]. The simplex has usually found a minimum. However, it may be trapped near some unusual topological feature, perhaps moving up and down a valley where coefficient values change but the function value does not. When dealing with new, unfamiliar, complicated functions, it is always advisable to experiment with initial estimates and step sizes and to examine at least the latter iterations in file ITER.OUT.

B. Results

In Figures 1, 2, and 3, test results begin after the lines "*****CALCULATING SIMPLEX*****". Tables 1 and 2 show the latter iterations from iteration history file ITER.OUT produced by executing the cases shown in Figures 1 and 2, respectively. After each execution, this file can be examined, printed, or renamed, but file ITER.OUT is destroyed and recreated near the beginning of each execution.

Figure 1 shows convergence criterion 1 being satisfied with 74 iterations for a selected error limit of 1.D-05. Table 1 shows that all coefficient relative errors are indeed less than 1.D-05 at iteration 74. This table also shows the construction of this diagnostic file, with results labeled for iteration number 69.

Figure 2 shows the same case as Figure 1 except that convergence criterion 2 was selected. Note that the final value of the function (0.414162E-24) is lower, and the number of iterations (224) is higher. Criterion 2 is often more stringent than criterion 1.

Table 2 shows iterations 211-224 from file ITER.OUT. Notice that beginning with iteration 213, the relative errors develop a pattern as iterations proceed: the first and third ones switch back and forth between pairs of values, and the second remains constant. This continues for $3*(MM+1)=12$ iterations, and then criterion 2 is satisfied. This type of error behavior is common when the simplex has locked in to a minimum. Given the logical rules for changing vertex values and the truncation/round-off limits in the computer's arithmetics, the program is unable to find a change which results in a lower function value. Without the logic for criterion 2, iterations would have proceeded until the iteration limit was exceeded.

The case in Figure 3 is the same as that in Figure 2 except the initial estimates were 200, 300, 400 instead of 2, 3, 4, which were much closer to the correct answers 1, 2, 3. Using convergence criterion 2, 268 iterations were required. Approximate wall clock execution time was 30 seconds for Figure 3 and 26 seconds for Figure 2 (224 iterations). For comparison, using single precision, the execution times were approximately 16 and 12 seconds, respectively. However, the estimation of the true function minimum of 0.0 was about $10^{-12} - 10^{13}$ using single precision, while double precision improved it to approximately $10^{29} - 10^{30}$.

IV. CONCLUSIONS

The computer program SIMMIN is capable of finding a minimum (if one exists) for any function which can be described using Fortran. It was tested on an IBM PC with an 8087 math coprocessor using Ryan-McFarland Fortran. This

simplex method determines values of initially unknown coefficients which result in a local function minimum.

The user can vary initial estimates of the coefficients, step sizes, maximum number of iterations, and the convergence criteria, of which two are available. Final values for the coefficients, an estimate of the function minimum, and the number of iterations required are displayed on the screen. A diagnostic iteration history is recorded on a separate file which can be examined after each execution.

The program can easily be converted for use on larger computers with larger word sizes, perhaps allowing the use of single precision. Although divergence is impossible using this simplex method, a price is often paid in more iterations and computing time than with some other traditional minimization methods, such as those mentioned in the introduction.

The user is cautioned against drawing overly optimistic conclusions from the test results. A very simple test function was used. Actual functions encountered in research and development - analytical, empirical, or hybrids - which cause problems when using minimization methods such as those mentioned in the introduction could require care when using simplex techniques as well.

Here are a few advantages simplex methods such as those in SIMMIN offer over other methods:

1. No derivatives of any kind are calculated.
2. Divergence is impossible.
3. Initial estimates for the coefficients can be poor, but this will slow down convergence, depending upon step size.
4. Simplex logic insures economical calculation of the response function.
5. No matrix operations are involved.
6. Constraint conditions which can be stated in Fortran can be added very easily.

Here are a few disadvantages of simplex compared with other methods:

1. More iterations and computing time can be required. SIMMIN is recommended for larger computers, especially when the function is complicated, when other methods have proved unsatisfactory or need to be verified, or when the user has little knowledge of the functional behavior.
2. The following can cause computational problems: very large initial estimates coupled with small steps, insensitivity of the function to one or more coefficients, too large an acceptable error when using an error comparison convergence criterion. These and other problem areas are discussed in much more detail in Reference 2.

REFERENCES

1. Nelder, J. A., Mead, R., "A Simplex Method for Function Minimization", Computer Journal, Vol. 7, pp. 308-313, 1965.
2. Caceci, Marco S., Cacheris, William P., "Fitting Curves to Data", Byte Magazine, pp. 340-362, May 1984.
3. Holloman, Miles E., Otto, William F., "PCFIT2 - General Purpose Curve Fitting Program for IBM PC", MICOM Technical Report RH-85-3, August 1985.
4. Holloman, Miles E., Otto, William F., Smith, Jerry D., "PCFIT Version 2.0 Plotting, Fitting, Interpolating Utility Program for IBM PC", MICOM Technical Report RD-DE-86-3, July 1986.

ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED -

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED. PROGRAM USUALLY CONVERGES WITHIN $20*(MM**2)$ ITERATIONS. -

ENTER MM INITIAL ESTIMATES OF PARAMETERS. BE SURE THE ENTRY ORDER CONFORMS WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC. ESTIMATES USUALLY CAN BE QUITE INACCURATE. -

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5. (STEP = ESTIMATE * MULTIPLIER) -

ENTER CONVERGENCE CRITERION OPTION -

- 1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.
- 2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE FUNCTION EXCEEDS $3*(MM+1)$ -

ENTER LIMIT ON RELATIVE ERRORS, THE SAME FOR ALL COEFFICIENTS. RECOMMENDED RANGE 1.D-06 to 1.D-04. -

***** CALCULATING SIMPLEX*****

FINAL VALUES OF COEFFICIENTS -

1.00000 2.00000 3.00000

FINAL VALUE (MINIMUM) OF FUNCTION = 0.417040E-10

NUMBER OF ITERATIONS = 74

Execution terminated : OK

Figure 1. Screen input and results for a typical execution of SIMMIN.
Convergence criterion 1 was selected.

TABLE 1. Latter Contents of Iteration History File ITER.OUT Produced During Execution With Input Parameters Shown in Figure 1.

69 = Iteration number

1.000

2.000 Current values of coefficients

3.000

0.1210E-09 Estimate of function value, printed only when the iteration led to a better (lower) function value.

1.48453975763265370E-05 7.86123165807031016E-06 7.56428230863572653E-06
70

Estimates of relative errors for the coefficients, MM = 3

1.0000

2.000

3.000

0.7947E-10

1.48453976763265370E-05 4.89503933556161228E-06 7.56428230863572653E-06
71

1.000

2.000

3.000

1.44438985795023159E-05 4.89503933556161228E-06 7.56428230863572653E-06
72

1.0000

2.000

3.000

0.3374E-10

1.20481762607656796E-05 3.12815572995717666E-06 5.20121882225269973E-06
73

1.000

2.000

3.000

0.3160E-10

1.20481762607656796E-05 3.12815572995717666E-06 2.91527801458411470E-06
74

1.000

2.000

3.000

All relative errors < 1.D-05.
Option 1 convergence (ICON = 1) obtained.

6.93297636726561162E-06 3.12815572995717666E-06 2.91527801458411470E-06

ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED -

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED. PROGRAM USUALLY CONVERGES WITHIN $20*(MM**2)$ ITERATIONS. -

ENTER MM INITIAL ESTIMATES OF PARAMETERS. BE SURE THE ENTRY ORDER CONFORMS WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC. ESTIMATES USUALLY CAN BE QUITE INACCURATE. -

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5. (STEP = ESTIMATE * MULTIPLIER) -

ENTER CONVERGENCE CRITERION OPTION -

1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.

2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE FUNCTION EXCEEDS $3*(MM+1)$ -

***** CALCULATING SIMPLEX*****

FINAL VALUES OF COEFFICIENTS -

1.00000 2.00000 3.00000

FINAL VALUE (MINIMUM) OF FUNCTION = 0.414162E-29

NUMBER OF ITERATIONS = 224

Execution terminated : OK

Figure 2. Screen input and results for a typical execution of SIMMIN.
Convergence criterion 2 was selected.

TABLE 2. Latter Contents of File ITER.OUT Produced by Input Parameters in Figure 2. Repetitive Pattern in Coefficient Errors Begins at Iteration 213.

211

1.000
2.000
3.000
0.4930E-31

2.22044604925031111E-15 1.11022302462515536E-15 8.88178419700124936E-16
212

1.0000
2.000
3.000

2.22044604925031111E-15 6.66133814775093530E-16 5.92118946466750122E-16
213

1.0000
2.000
3.000

1.99840144432528138E-15 6.66133814775093530E-16 5.92118946466750023E-16
214

1.000
2.000
3.000

2.22044604925031111E-15 6.66133814776093530E-16 5.92118946466750122E-16
215

1.0000
2.000
3.000

1.99840144432528138E-15 6.66133814775093530E-16 5.92118946466750023E-16
216

1.000
2.000
3.000

2.22044604825031111E-15 6.66133814775093530E-16 5.92118946466750122E-16
217

1.0000
2.000
3.000

TABLE 2. (Concluded)

1.99840144432528138E-15 218	6.66133814775093530E-16	5.92118946466750023E-16
1.000		
2.000		
3.000		
2.22044604925031111E-15 219	6.66133814775093530E-16	5.92118946466750122E-16
1.0000		
2.000		
3.000		
1.99840144432528138E-15 220	6.66133814775093530E-16	5.92118946466750023E-16
1.000		
2.000		
3.000		
2.22044604925031111E-15 221	6.66133814775093530E-16	5.92118946466750122E-16
1.0000		
2.000		
3.000		
1.99840144432528138E-15 222	6.66133814775093530E-16	5.92118946466750023E-16
1.000		
2.000		
3.000		
2.22044604925031111E-15 223	6.66133814775093530E-16	5.92118946466750122E-16
1.0000		
2.000		
3.000		
1.99840144432528138E-15 224	6.66133814775093530E-16	5.92118946466750023E-16
1.000		
2.000		
3.000		
2.22044604925031111E-15	6.66133814775093530E-16	5.92118946466750122E-16

ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED - 3

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED.
PROGRAM USUALLY CONVERGES WITHIN $20*(MM**2)$ ITERATIONS. - 300

ENTER MM INITIAL ESTIMATES OF PARAMETERS. BE SURE THE ENTRY ORDER CONFORMS
WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC. ESTIMATES
USUALLY CAN BE QUITE INACCURATE. -
200,300,400

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5. (STEP =
ESTIMATE * MULTIPLIER) - .2

ENTER CONVERGENCE CRITERION OPTION -

1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.

2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE
FUNCTION EXCEEDS $3*(MM+1)$ - 2

***** CALCULATING SIMPLEX*****

FINAL VALUES OF COEFFICIENTS -

1.00000 2.00000 3.00000

FINAL VALUE (MINIMUM) OF FUNCTION = 0.246519E-30

NUMBER OF ITERATIONS = 268

Execution terminated : OK

Figure 3. Screen input and results for a typical execution of SIMMIN.
Initial coefficient estimates were 100, 200, 300.
Convergence criterion 2 was selected.

APPENDIX
FORTRAN SOURCE CODE FOR SIMMIN

```

1          PROGRAM SIMMIN
2 C
3 C      PROGRAM SIMMIN CAN FIND A MINIMUM (IF ONE EXISTS) FOR ANY
4 C      FUNCTION DEFINED BY THE USER IN SUBROUTINE FUNC. THE FUNCTION
5 C      CAN BE LINEAR OR NONLINEAR, PARTIALLY EMPIRICALLY DEFINED, AND
6 C      NEED NOT BE CONTINUOUS. SIMMIN DOES THIS BY USING SIMPLEX
7 C      TECHNIQUES TO CALCULATE THE VALUES OF INITIALLY UNKNOWN
8 C      COEFFICIENTS THAT YIELD THE MINIMUM. PROGRAM IS BASED ON A
9 C      PASCAL ALGORITHM IN "FITTING CURVES TO DATA", BYTE, MAY 1984,
10 C     WHICH WAS CONVERTED TO FORTRAN77 AND MODIFIED FOR USE IN
11 C     GENERAL PC CURVE FITTING PROGRAMS BY MILES HOLLOMAN AND BILL OTTO,
12 C     DIRECTED ENERGY DIRECTORATE, MICOM. FURTHER MODIFICATIONS
13 C     RESULTED IN THIS INITIAL VERSION OF SIMMIN BY JERRY D. SMITH,
14 C     OF THE SAME DIRECTORATE. USER SHOULD BE FAMILIAR ENOUGH WITH
15 C     FORTRAN TO MODIFY SUBROUTINE FUNC AS REQUIRED, CONVERGENCE
16 C     CRITERIONS IF DESIRED (SEE COMMENTS WITHIN SUBROUTINE SIMPLX),
17 C     AND PERHAPS MINOR FORMAT CHANGES IF MANY UNKNOWN COEFFICIENTS
18 C     COEF(I) OCCUR IN THE FUNCTION. CURRENT MAX. NO. = 10.
19 C     NOTE --- EXECUTION TIMES INCREASE GREATLY WITH NUMBER OF
20 C     COEFFICIENTS, ABOVE THREE.
21 C
22 C     AFTER EXECUTION FILE ITER.OUT WILL CONTAIN A HISTORY OF THE
23 C     ITERATION RESULTS : COEFFICIENT AND FUNCTION VALUES, AND RELATIVE
24 C     ERROR VALUES FOR THE COEFFICIENTS. USER CAN EXAMINE AFTER EXECU-
25 C     TION IF DESIRED. IT IS DESTROYED AND RECREATED DURING THE NEXT
26 C     EXECUTION.
27 C
28 C     ADVANTAGES OVER OTHER NONLINEAR METHODS:
29 C     1. NO DERIVATIVES ARE CALCULATED.
30 C     2. DIVERGENCE IS IMPOSSIBLE.
31 C     3. INITIAL ESTIMATES FOR COEFFICIENTS CAN BE QUITE POOR,
32 C     ALTHOUGH THIS WILL SLOW DOWN CONVERGENCE.
33 C     4. SIMPLEX LOGIC INSURES ECONOMICAL CALCULATIONS OF RESPONSE
34 C     FUNCTION.
35 C     5. NO MATRIX OPERATION IS INVOLVED.
36 C     6. CONSTRAINTS CAN BE ADDED VERY EASILY. (SEE SUBROUTINE FUNC.)
37 C
38 C     DISADVANTAGES OVER OTHER NONLINEAR METHODS:
39 C     1. MORE ITERATIONS AND COMPUTING TIME ARE OFTEN REQUIRED. SIMMIN
40 C     RECOMMENDED FOR FAST PCs, AND NUMBER-CRUNCHERS, ESPECIALLY
41 C     WHEN THE FUNCTION TO BE MINIMIZED IS COMPLICATED AND USER HAS
42 C     VERY LITTLE IF ANY KNOWLEDGE OF THE FUNCTIONAL BEHAVIOR.
43 C     2. FOLLOWING CAN CAUSE COMPUTATIONAL PROBLEMS, AND RESULT IN
44 C     INACCURATE ANSWERS: VERY LARGE INITIAL GUESSES, VERY SMALL
45 C     INITIAL INCREMENTS, INSENSITIVITY OF FUNCTION TO ONE OR MORE
46 C     COEFFICIENTS, TOO LARGE AN ACCEPTABLE ERROR (WHEN USING
47 C     CONVERGENCE OPTION ICON=2).
48 C
49 C
50 C          JERRY D. SMITH
51 C          11 MAY 87
52 C
53 C
54 C     COMMON FINAL(11),SIMP(11,11),STEP(11),MM,LUO,MAXITR,NITER,ICON,RX
55 C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
56 C

```

```

57      LUO=7
58      OPEN(UNIT=LUO,FILE='ITER.OUT',STATUS='UNKNOWN')
59      CLOSE(UNIT=LUO,STATUS='DELETE')
60      OPEN(UNIT=LUO,FILE='ITER.OUT',STATUS='NEW')
61 C
62      WRITE(*,1600)
63 1600  FORMAT(1H1,'ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED - ')
64      READ(*,*) MM
65      WRITE(*,1601)
66 1601  FORMAT(1X,'ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVER
67      XAL HUNDRED.',/' PROGRAM USUALLY CONVERGES WITHIN 20*(MM+2) ITERAT
68      XIONS. - ')
69      READ(*,*) MAXITR
70      MM1=MM+1
71      WRITE(*,1602)
72 1602  FORMAT(1X,'ENTER MM INITIAL ESTIMATES OF PARAMETERS. BE SURE'
73      X,/, ' THE ENTRY ORDER CONFORMS WITH YOUR DEFINITION OF PARAMETERS'
74      X,/, ' COEF(I) IN SUBROUTINE FUNC. ESTIMATES USUALLY CAN BE QUITE'
75      X,/, ' INACCURATE - /')
76      READ(*,*) (SIMP(I,1),I=1,MM)
77      WRITE(*,1603)
78 1603  FORMAT(1X,'ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1
79      X' /, ' TO 0.5. (STEP=ESTIMATE * MULTIPLIER) - ')
80      READ(*,*) SMULT
81      DO 10 J=1,MM
82 10    STEP(J)=SIMP(J,1)*SMULT
83 C
84      PRINT 1610
85 1610  FORMAT(1X, 'ENTER CONVERGENCE CRITERION OPTION -',/1X, '1, FOR COM
86      XPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.',/1X, '2, FOR THE CON
87      XDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO',/1X, ' DECRE
88      XASE THE FUNCTION EXCEEDS 3*(MM+1) - ')
89      READ(*,*) ICON
90      IF(ICON.EQ.1) THEN
91      PRINT 1612
92 1612  FORMAT(1X, 'ENTER LIMIT ON RELATIVE ERRORS, THE SAME FOR ALL COEFF
93      XICIENTS. ',/1X, 'RECOMMENDED RANGE 1.D-05 TO 1.D-04 -')
94      READ(*,*) RX
95      ENDIF
96 C
97      CALL SIMPLEX
98 C
99      WRITE(*,1604) (FINAL(I),I=1,MM)
100 1604  FORMAT(1X,'FINAL VALUES OF COEFFICIENTS - ',/3X, 6G12.6)
101      PRINT 1606, FINAL(MM1)
102 1606  FORMAT( ' FINAL VALUE (MINIMUM) OF FUNCTION = ', 612.6)
103      PRINT 1608, NITER
104 1608  FORMAT(1X,'NUMBER OF ITERATIONS=',I4)
105 C
106      STOP 'OK'
107      END

```

```

108     SUBROUTINE SIMPLEX
109 C
110     LOGICAL DONE,LEQ
111     COMMON FINAL(11),SIMP(11,11),STEP(11),MM,LUO,MAXITR,NITER,ICGN,RX
112     DIMENSION CENTER(11),ERROR(11),RMAXER(11),PS(11),Q(11),IH(11)
113     DIMENSION RNEXT(11),L(11)
114     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
115 C
116     DLOW=1.0D+38
117     DLOWL=1.0D+38
118     LCNT=0
119     PRINT 1030
120 1030  FORMAT(' ***** CALCULATING SIMPLEX ***** ')
121 C
122     MMI=MM+1
123     DO 1040 I=1,11
124 1040  RMAXER(I)=RX
125 C
126 C     FOLLOWING ARE PARAMETERS TO VARY THE SIMPLEX MODIFICATIONS AND
127 C     SHOULD GENERALLY BE ACCEPTED AS OPTIMAL VALUES.
128 C
129     ALFA = 1.0
130     BETA = 0.5
131     GAMMA = 2.0
132 C
133 C
134     TWO = 2.0
135     ROOT2 = DSQRT(TWO)
136     CALL FUNC(SIMP(1,1),MM1)
137     DO 1060 I=1,MM
138     RNN = MMI
139     PS(I)=STEP(I)*((SQRT(RNN)+MM-1))/(MM*ROOT2)
140     Q(I)=STEP(I)*((SQRT(RNN))-1)/(MM*ROOT2)
141 1060  CONTINUE
142     DO 1080 I=2,MM1
143     DO 1070 J=1,MM
144 1070  SIMP(J,I)=SIMP(J,I)+Q(J)
145     SIMP(I-1,I)=SIMP(I-1,I)+PS(I-1)
146     CALL FUNC(SIMP(I,I),MM1)
147 1080  CONTINUE
148     DO 1090 I=1,MM1
149     L(I)=1
150 1090  IH(I)=1
151     CALL ORDER(MM1,SIMP,L,IH)
152     NITER=0
153 1100  DONE=.TRUE.
154     NITER=NITER+1
155 C
156 C     COMPUTE CENTROID OF THE SIMPLEX.
157 C
158     DO 1110 I=1,MM1
159 1110  CENTER(I)=0.0
160     DO 1130 I=1,MM1
161     IF(I.EQ.IH(MM1)) GO TO 1130
162     DO 1120 J=1,MM
163     CENTER(J)=CENTER(J)+SIMP(J,I)

```

```

164 1120 CONTINUE
165 1130 CONTINUE
166 C
167 C TRY SPECULAR REFLECTION.
168 C
169 DO 1140 I=1,MM1
170 CENTER(I)=CENTER(I)/MM
171 RNEXT(I)=(1.0+ALFA)*CENTER(I)-ALFA*SIMP(I,IH(MM1))
172 1140 CONTINUE
173 CALL FUNC(RNEXT,MM1)
174 C
175 IF(RNEXT(MM1).LE.SIMP(MM1,L(MM1))) GO TO 1150
176 GO TO 1170
177 1150 CALL NEWUR(MM1,SIMP,RNEXT,IH)
178 C
179 C TRY AN EXPANSION.
180 C
181 DO 1160 I=1,MM
182 RNEXT(I)=GAMMA*SIMP(I,IH(MM1))+(1.0-GAMMA)*CENTER(I)
183 1160 CONTINUE
184 CALL FUNC(RNEXT,MM1)
185 C
186 IF(RNEXT(MM1).LE.SIMP(MM1,L(MM1)))
187 X CALL NEWUR(MM1,SIMP,RNEXT,IH)
188 GO TO 1250
189 C
190 1170 IF(RNEXT(MM1).LE.SIMP(MM1,IH(MM1))) GO TO 1180
191 GO TO 1190
192 1180 CALL NEWUR(MM1,SIMP,RNEXT,IH)
193 GO TO 1250
194 C
195 C TRY A CONTRACTION.
196 C
197 1190 DO 1200 I=1,MM
198 1200 RNEXT(I)=BETA*SIMP(I,IH(MM1))+(1.0-BETA)*CENTER(I)
199 CALL FUNC(RNEXT,MM1)
200 C
201 IF(RNEXT(MM1).LE.SIMP(MM1,IH(MM1))) GO TO 1210
202 GO TO 1220
203 1210 CALL NEWUR(MM1,SIMP,RNEXT,IH)
204 GO TO 1250
205 C
206 C DO A SHRINKAGE.
207 C
208 1220 DO 1240 I=1,MM1
209 DO 1230 J=1,MM
210 1230 SIMP(J,I)=SIMP(J,I)+SIMP(J,L(MM1))*BETA
211 1240 CALL FUNC(RNEXT,MM1)
212 1250 CONTINUE
213 1270 FORMAT(I4)
214 1280 FORMAT(6I2.4)
215 IF(.LEQ) LCNT=LCNT+1
216 IF(.NOT.LEQ) LCNT=0
217 DLQWL=DLOW
218 C
219 WRITE(LUQ,1270) NITER

```

```

220      DO 1308 I=1,MM1
221      IF(I.NE.MM1) WRITE(LUO,1280) RNEXT(I)
222      IF(I.EQ.MM1.AND.RNEXT(MM1).LT.DLOW) WRITE(LUO,1280) RNEXT(I)
223 1308  FINAL(I)=RNEXT(I)
224      WRITE(LUO,1420)
225      IF(RNEXT(MM1).LT.DLOW) DLOW=RNEXT(MM1)
225 C
227 1305  LEQ=(DLOWL.EQ.DLOW)
229      CALL ORDER(MM1,SIMP,L,IH)
229 C
230      DO 1320 J=1,MM
231      ERROR(J)=(SIMP(J,IH(J))-SIMP(J,L(J)))/SIMP(J,IH(J))
232 1320  CONTINUE
233      WRITE(LUO,*) (ERROR(J),J=1,MM)
234 C
235 C      FOLLOWING IS PRESENT AUTHORS' CRITERION FOR CONVERGENCE,
236 C      BASED ON EXPERIENCE WITH NONLINEAR FUNCTIONS IN FITTING
237 C      APPLICATIONS. ESSENTIALLY, THIS CRITERION SAYS THAT IF NUMBER OF
238 C      CONSECUTIVE UNSUCCESSFUL ATTEMPTS FIND A NEW DIRECTION AND
239 C      SHAPE TO DECREASE THE VALUE OF THE FUNCTION REACHES MM1*3,
240 C      THEN CALL IT A GAME AND RETURN TO MAIN PROGRAM. THE SIMPLEX HAS
241 C      EITHER FOUND A MINIMUM AND IS SPINNING ITS WHEELS, OR IT IS EATING
242 C      UP TIME DUE TO POORLY CHOSEN INPUTS. TRY AGAIN WITH DIFFERENT
243 C      INITIAL ESTIMATES AND STEP SIZES, IF USER SUSPECTS SOMETHING
244 C      UNUSUAL. IN FACT, IT IS ALWAYS GOOD PROCEDURE TO VERIFY
245 C      RESULTS BY REPUNNING WITH DIFFERENT INPUTS, AND TO INSPECT
246 C      ITERATION HISTORY FILE ITER.OUT.
247 C
248      IF(ICON.EQ.2 .AND. LCNT.GE.MM1*3) GO TO 1350
249 C
250 C      THE FOLLOWING IS A DIFFERENT TYPE OF CONVERGENCE CRITERION.
251 C
252      IF(ICON.EQ.1) THEN
253      K=0
254      DO 1322 J=1,MM
255      IF(ERROR(J).LT.RMAXER(J)) K=K+1
256 1322  CONTINUE
257      IF(K.GE.MM) GO TO 1350
258      ENDIF
259      IF(NITER.GT.MAXITR) GO TO 1350
260 C
261 C      TRY AGAIN.
262 C
263      GO TO 1100
264 C
265 C      CONVERGENCE OBTAINED, OR ITERATION LIMIT EXCEEDED, ETC.
266 C
267 1350  CONTINUE
268      WRITE(LUO,1420)
269 1420  FORMAT(' ')
270      RETURN
271      END

```

```

272      SUBROUTINE FUNC(COEF,MM1)
273 C
274 C      A SIMPLE TEST FUNCTION, WITH THREE UNKNOWN COEFFICIENTS
275 C      COEF(J), J=1,3, IS TO BE MINIMIZED. NOTE THE CURRENT FUNCTION
276 C      VALUE (THE MM+1 th VERTEX OF THE SIMPLEX) IS STORED IN COEF(MM+1)
277 C
278      DIMENSION COEF(11)
279      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
280 C
281      TERM1=COEF(1)-1.0
282      TERM2=COEF(2)-2.0
283      TERM3=COEF(3)-3.0
284      COEF(MM1)=TERM1**2 + TERM2**2 + TERM3**2
285      RETURN
286 C
287 C      CONSTRAINT CONDITION EXCEEDED - SEND BACK A MESSAGE TO
288 C      SUBROUTINE SIMPLX, BY TAGGING ON A LARGE VALUE TO THE FUNCTION.
289 C      NOTE ---- USER MUST PROTECT AGAINST SUCH OCCUPRENCES AS TRYING
290 C      TO TAKE LOGARITHMS OF 0 OR NEGATIVE NUMBERS, ETC. TREAT THEM
291 C      AS CONSTRAINTS, AND DIRECT PROGRAM TO STATEMENT 999, WHEN
292 C      NECESSARY.
293 C
294 999   CONTINUE
295      COEF(MM1)=COEF(MM1)+1.030
296      RETURN
297      END

```

```
298 C
299 SUBROUTINE NEWUR(MM1,SIMP,RNEXT,IH)
300 C
301 C THIS SUBROUTINE STORES THE NEW CHOICES FOR VERTICES, WHICH HAVE
302 C RESULTED IN A DECREASE IN THE VALUE OF THE RESPONSE FUNTION.
303 C
304 DIMENSION SIMP(11,1),RNEXT(1),IH(1)
305 IMPLICIT DOUBLE PRECISION (A-H,O-Z)
306 C
307 DO 3010 I=1,MM1
308 SIMP(I,IH(MM1))=RNEXT(I)
309 3010 CONTINUE
310 RETURN
311 END
```

```

312 Q
313 SUBROUTINE ORDER(MM1,SIMP,L,IH)
314 C
315 C THIS SUBROUTINE DETERMINES CURRENT LOWEST AND HIGHEST VERTICES,
316 C AND STORES ORDER INFORMATION USING SECOND INDEX OF SIMP(J,I).
317 C THIS WILL BE USED BY SUBROUTINE SIMPLX IN DETERMINING THE NEXT
318 C TRIAL SHAPE AND LOCATION OF THE SIMPLEX.
319 C
320 DIMENSION SIMP(11,1),L(1),IH(1)
321 IMPLICIT DOUBLE PRECISION (A-H,O-Z)
322 C
323 DO 4020 J=1,MM1
324 DO 4010 I=1,MM1
325 IF(SIMP(J,I).LT.SIMP(J,L(J))) L(J)=I
326 IF(SIMP(J,I).GT.SIMP(J,IH(J))) IH(J)=I
327 4010 CONTINUE
328 4020 CONTINUE
329 RETURN
330 END

```

DISTRIBUTION

	<u>No. of Copies</u>
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
US Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
AMSMI-RD, Dr. McCorkle Dr. Rhoades	1
AMSMI-RD-CS-T, Record Copy	1
AMSMI-RD-CS-R, Reference	15
AMSMI-RD-DE, Jerry D. Smith	50
AMSMI-GC-IP, Mr. Bush	1

END

DATE

FILMED

8-88

DTIC