N69-35074
NASA CR 104133

A GRID SEARCH OPTIMIZATION SUBROUTINE FOR USE WITH THE GOSPEL OPTIMIZATION SOFTWARE PACKAGE

Prepared under Grant NGL-03-002-136 for the
Instrumentation Division of the Ames Research Center
National Aeronautics and Space Administration

by

L. P. Huelsman

and

G. R. Allgaier

ENGINEERING EXPERIMENT STATION
COLLEGE OF ENGINEERING
THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA

A GRID SEARCH OPTIMIZATION SUBROUTINE FOR USE WITH THE GOSPEL OPTIMIZATION SOFTWARE PACKAGE

Prepared under Grant NGL-03-002-136 for the
Instrumentation Division of the Ames Research Center
National Aeronautics and Space Administration

by

L. P. Huelsman
and
G. R. Allgaier

Department of Electrical Engineering
University of Arizona
Tucson, Arizona

Abstract: This report describes a subroutine implementing an optimization strategy which supplements those described in an earlier report.[1] The strategy is a grid search. The results obtained from applying this strategy to a pair of test problems are discussed.

June 1969

TABLE OF CONTENTS

I. INTRODUCTION

This is one of a series of reports concerning the use of digital computational techniques in the analysis and synthesis of DLA (Distributed-Lumped-Active) networks. This class of networks consists of three distinct types of elements, namely distributed elements (modeled by partial differential equations), lumped elements (modeled by algebraic equations and ordinary differential equations), and active elements (modeled by algebraic equations). Such a characterization is especially applicable to the broad class of circuits referred to as linear integrated circuits, since the required fabrication techniques readily produce elements which may be referred to as "distributed", as well as producing elements which may be characterized as "lumped" and/or "active". The DLA class of networks is capable of realizing network functions with a wide range of properties. In addition, such realizations usually have fewer components and superior characteristics than realizations using only lumped elements, or realizations using lumped elements and active elements. The analysis problem for this class of network, however, is considerably more complex than the analysis problem for more restricted classes of networks. The synthesis problem is even more challenging, and the results achieved to date have been far from general.

One of the more promising approaches to the synthesis problem appears to be the use of optimization techniques. The experience of research workers in this field has indicated that in order to successfully apply optimization techniques to a wide range of problems, it is desirable to have available a varied collection of optimization strategies. To be fully useful, the individual strategies of such a collection must be so designed that any one of them can be applied to the same problem, without requiring that the problem be modified. Thus, the individual optimization strategies can be considered as forming the elements of an optimization software package, in which various logical decisions can be incorporated as an "executive monitor" to successfully apply the different strategies in such a way as to obtain the best final results.

In a previous report the formulation of general problem structure and the development and testing of digital computer program incorporating a series of optimization strategies was described.[1] These strategies include such well known techniques as: random grid search, random direction and step size search, steepest descent, Newton-Raphson, and Fletcher-Powell. The program was named GOSPEL (for General Optimization Software Program for ELectrical Networks). In this report the development of one additional optimization strategy is discussed. This is a (non-random) grid search optimization strategy. It is named OPT1. For conciseness, the material contained in the original report describing the general problem structure and the test problems has not been duplicated in this report. Thus, the reader should refer to the original GOSPEL report for the background material pertinent to the development contained in this report.

II. THE GRID SEARCH OPTIMIZATION STRATEGY

A grid search optimization strategy is a systematic testing of an entire range of values for a set of n parameters. These parameter values are determined by dividing the range of interest of each parameter into equal segments. Thus an initial range of values must be specified as well as the <u>number</u> of values to be examined for each parameter. The grid search optimization strategy then procedes by examining all possible combinations of these parameter values and stores that combination which comes closest to meeting the design criterion. Several variations of this basic procedure are possible. For example, instead of storing just the single set of parameter values which gave the best result, it may be desirable to store several of the best sets of parameter values in order to make available a wide choice of starting points for use in other optimization strategies. Another possible variation is to set up a smaller range of values surrounding the best set of parameter values found in searching the original grid, and then make a further search of this reduced area.

A grid search of the type described above is not, in general, useful for an exact determination of a minimum of a specified error function. This is because (1) a large number of parameters may result in an impossibly large number of cases to be analyzed; (2) if the grid spacings are too large, it is possible that the program may completely miss a global minimum and find only local minima; and (3) if the grid spacings are too small the computation time may become excessive. Nevertheless, because the grid search optimization strategy does sample a large volume of parameter space, the grid search routine is useful in preliminary studies to determine the general nature of the topology of a given problem.

III. GRID SEARCH OPTIMIZATION SUBROUTINE OPT1

In this section, a subroutine named OPT1 which was written to implement a grid search optimization strategy is described. The options which are available, and the additional subroutines required by the optimization strategy are described below.

Options

1. This option permits a decision as to whether to store more than one "best" value. The controlling variables for this option are NOPT (1,1) and PARAM(1,1). If the option is taken [NOPT(1,1) = 1], the value of PARAM(1,1) is read to determine the number of "best" values to be computed and stored. Subroutine ORDER (see below) is then implemented to order and store these values. PARAM(1,1) may be set for any value from 2 to 10. If a value is not read-in for this parameter, it is initialized to a value of 5 by the subroutine. If the option is not taken [NOPT(1,1) = 0], then only the single best set of parameters will be computed and stored.
2. This option permits a decision as to whether an extended printout of all the points sampled is to be made. The controlling variable is NOPT(1,2). If the option is taken [NOPT(1,2) = 1], the subroutine will print the results of each trial thus permitting an evaluation of the topology of the entire region to be made. In addition, the best single result [NOPT(1,1) = 0] or an ordered set of several of the best results [NOPT(1,1) = 1] will be printed. If the option is not taken, [NOPT(1,2) = 0], only the printout specified by NOPT(1,1) will be made.
3. This option permits a decision to make a local search around the best point previously found. The controlling variable for this option is NOPT(1,3). If the option is taken [NOPT(1,3) = 1], the range of the search area as defined by the variables XU(I) and

XL(I) is reduced to a value of one-half the former step size on either side of the parameter values which determine the best point previously found. The search is then repeated on this reduced grid using the same number of values for each parameter. At the conclusion of this local search, this reduction process is again repeated. The reduction process will continue until the error is less than ERMIN or the number of iterations exceeds ITMAX. If this option is not taken /NOPT(1,3) = 0/, the program will terminate after searching the entire grid or when the number of iterations exceeds ITMAX.

Significant Variables

Some of the significant variables which are not part of the common array of variables listed in the original GOSPEL report are listed below.

YERRX(I,J) - an array which stores the I best values determined by the program. For J = 1, YERRX(I,J) stores the error associated with the Ith best set of parameter values. For the range of I from 2 to N + 1, YERRX(I,J) stores the values of the N parameters which specify the Ith best point.

XM(I,J) - an array which stores up to J values of each of the I parameters. These are the values which determine the grid points which are to be tested. J may have a different value for the various parameters. The maximum value of J which is permitted by the dimensioning of the program is 20. Thus, this is the maximum number of values of any one parameter which can be used.

NY - the number of the best values of the parameters which are to be stored and printed /this is equal to PARAM(1,1)/.

NC - the total number of combinations of sets of parameter values which are to be tested. This is equal to the product of the quantities KX(I) which specify the number of values of the Ith parameter (I = 1,2, ..., N).

Other Subroutines Used

There are three subroutines which are used in connection with the subroutine OPT1. These are:

SUBROUTINE COMB(K,A,NA,NV,AV) - This subroutine is used to produce all possible combinations of the parameter values. K is index for the combination number. A is a two-dimensional array of elements A(I,J) in which are stored the Jth value of the Ith parameter. NA is a one-dimensional array with element NA(I) which specify the number of values of the Ith parameter. NV is the number of parameter combinations. AV is a one-dimensional array with elements AV(I) giving the resulting vector of parameter values corresponding with the Kth combination.

SUBROUTINE ORDER(YERR, NY, YERRX, X) - This subroutine compares the value of the error YERR computed by the subroutine ERR (see the original GOSPEL report) for the current set of parameter values, and ranks this value of YERR in descending order in the YERRX array. NY such values of error are stored along with the associated parameter values.

SUBROUTINE REDUC - This subroutine reduces the range of parameter values to plus or minus one-half of the former step size for each parameter. This is done by computing new values of XU(I) and XL(I). All input and output of information to this subroutine is made through the labeled common array.

IV. EXAMPLES OF THE USE OF OPT1

The grid search subroutine OPT1 was applied to the test problems described in the GOSPEL report. The data for the various runs is summarized in Table 1. Some comments on these runs follow:

In test problem one, all runs gave a good error result, and, as the grid size was reduced, the error became smaller. The final parameter values for different grid sizes were considerably different from each other, however, indicating the importance of finer search gradations. The second and third lines of data shown in the table for each run resulted from local searches about the previous "best" parameter values. The values of XL(I) were 0.0 for all parameters and the values of XU(I) were 2.0 for all parameters. Although the final error was low, the total computer time required to obtain this error was quite high in comparison with that required by other optimization strategies. For such a comparison the reader is referred to the original GOSPEL report.

In test problem two, the same upper and lower bounds of X(I) were used for runs 1 and 2 as were used for problem one. For run 1 the range of X(I) was divided into grids of one-fifth. The final error was so large (17202) at the end of run 1 that, on run 2, the grid size was reduced to one-sixth to improve this error. The final error resulting from run 2 was smaller (11429) but still considered excessive. This is probably due to the fact that the function defined by the problem has

TABLE 1. RESULTS OF TEST PROBLEMS

The tabulation below gives the results of the various computer runs made on the two test problems described in the GOSPEL report.

RUN NO.	GRID SIZE	X(1)	X(2)	X(3)	X(4)	X(5)	FINAL ERROR	ITERA TIONS	RUN TIME ON 6400
		FINAL PARAMETER VALUES							
				TEST PROBLEM NUMBER ONE					
1	3	.6667	2.000	.6667	1.333	.6667	7.560	243	3 sec.
		.6667	1.852	.8333	1.260	.6667	1.039	486	6 sec.
		.6667	1.852	.8313	1.260	.6667	1.038	729	9 sec.
2	4	.5000	1.500	1.000	2.000	.5000	6.274	1024	11 sec.
		.5781	1.398	.945	1.852	.5781	.468	2048	22 sec.
		.6036	1.397	.944	1.849	.5687	.318	3072	33 sec.
3	5	.4000	.1200	1.200	1.200	1.200	.169	3125	33 sec.
		.4440	1.264	1.128	1.128	1.196	.043	6250	66 sec.
		.4388	1.263	1.126	1.126	1.199	.009	9375	99 sec.
				TEST PROBLEM NUMBER TWO					
1	5	.4000	.800	.400	1.200	2.000	21278	3125	22 sec.
		.3920	.940	.392	1.128	2.200	17333	6250	44 sec.
		.3909	.947	.391	1.127	2.242	17202	9375	66 sec.
2	6	.3333	1.000	.333	1.000	2.000	12018	7776	55 sec.
		.3264	.991	.326	.991	2.167	11429	15552	110 sec.
3	5*	.1000	1.100	.100	.900	1.080	22.88	3125	22 sec.
		.1016	1.103	.102	.898	1.080	13.70	6250	44 sec.
		.1022	1.103	.101	.898	1.080	13.50	9375	66 sec.

*The Third run had different upper and lower bounds on X(I) than the first run.

a very sharp minimum, and in general, a grid search optimization strategy is unable to locate sharp minimums. The third run used very narrow upper and lower limits (see below) in the general area determined by the results of other optimization strategies applied to the same problem. The final error was 13.50. The upper and lower limits used were:

XL(1) = 0.07	XU(1) = 0.12
XL(2) = 1.07	XU(2) = 1.12
XL(3) = 0.07	XU(3) = 0.12
XL(4) = 0.87	XU(4) = 0.92
XL(5) = 1.07	XU(5) = 1.12

As has been borne out by the test results, the grid search is unable to easily locate sharply defined optimum points. The large relative computing time is also evident, indicating the inefficiency of the method. The increase in this time as a function of the grid size is also readily observed to be large even when the grid increment size is only one-third or one-fourth of the total range of the individual parameter range.

V. CONCLUSION

In this report the basic theory and implementation of a non-random grid search optimization strategy has been discussed. This optimization strategy is so defined that it will function as a integral part of the GOSPEL optimization software package described in an earlier report. Although the grid search optimization strategy described herein is, in general, not useful for accurately finding precise minimums. Nevertheless, it does have considerable application. For example, it is useful when the researcher desires to methodically cover a large area of N-dimensional space so that he may develop some information on the contours followed by some error function. It is also useful for making preliminary studies to make a coarse evaluation of the topology of a problem about which relatively little information is available.

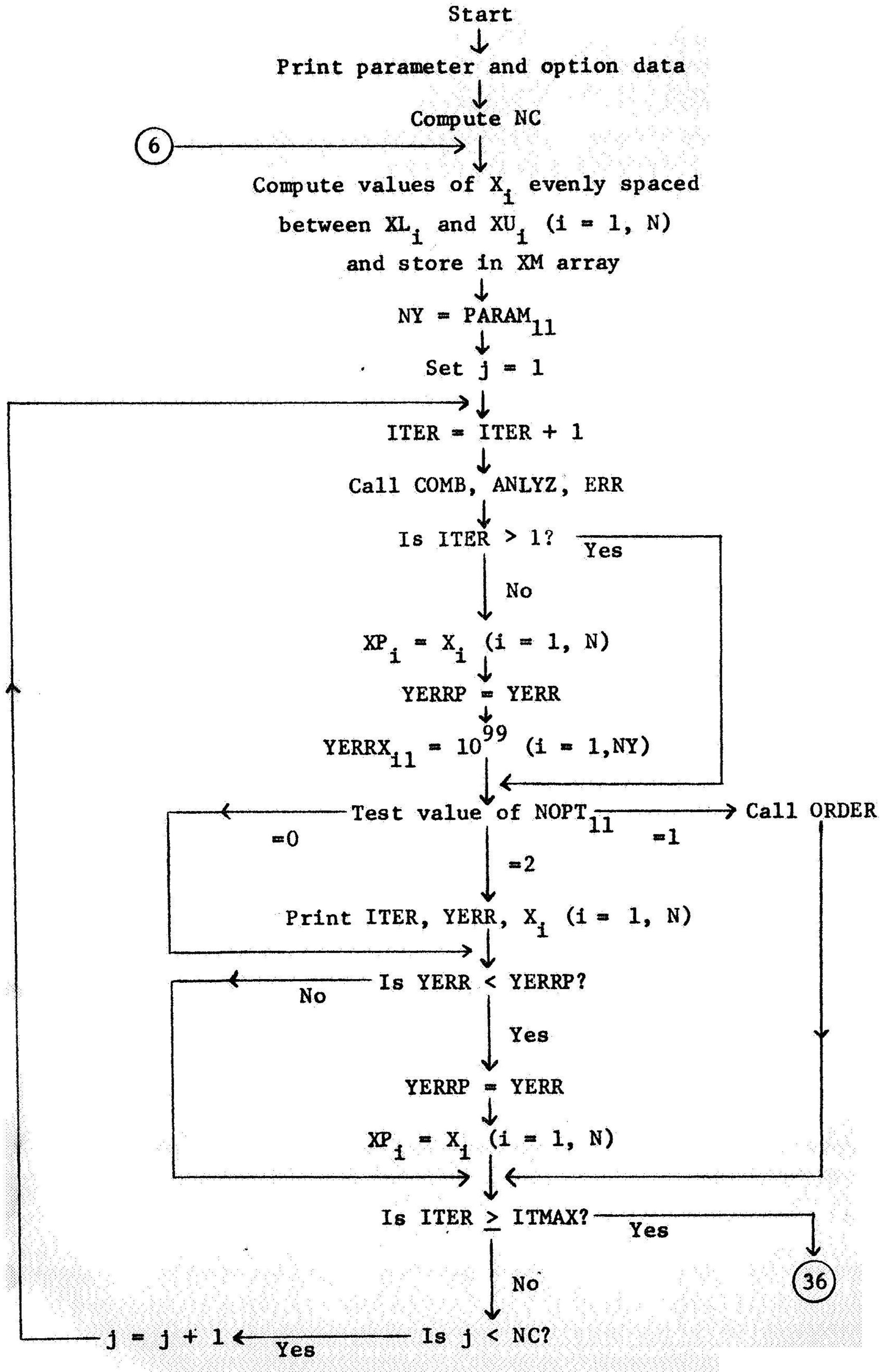
ACKNOWLEDGEMENT

The research described in this report was supported in part by the Instrumentation Division of the Ames Research Center of the National Aeronautics and Space Administration under Grant NGL-03-002-136. The author wishes to express his appreciation to the grantee, and especially to Dr. W. J. Kerwin, Chief, Electronics Research Branch, Instrumentation Division, NASA Ames Research Center, for the encouragement given to this research project. The author also wishes to acknowledge the assistance of Mr. Glen Allgaier, who assisted in the development of the program.

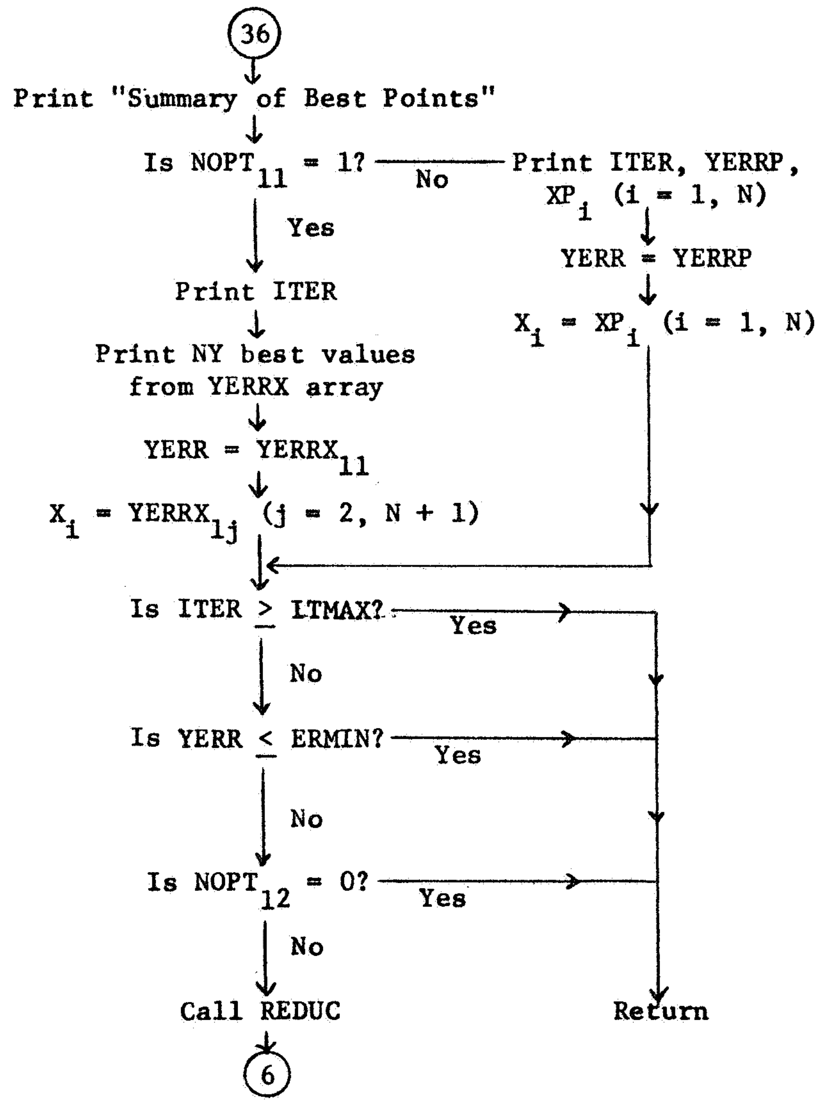
REFERENCES

1. "GOSPEL - A General Optimization Software Package for Electrical Network Design", by L. P. Huelsman, 94 pages, report prepared under NASA Grant NGL 03-002-136, Sept. 1968.

APPENDIX



Flow Chart for Subroutine OPT1

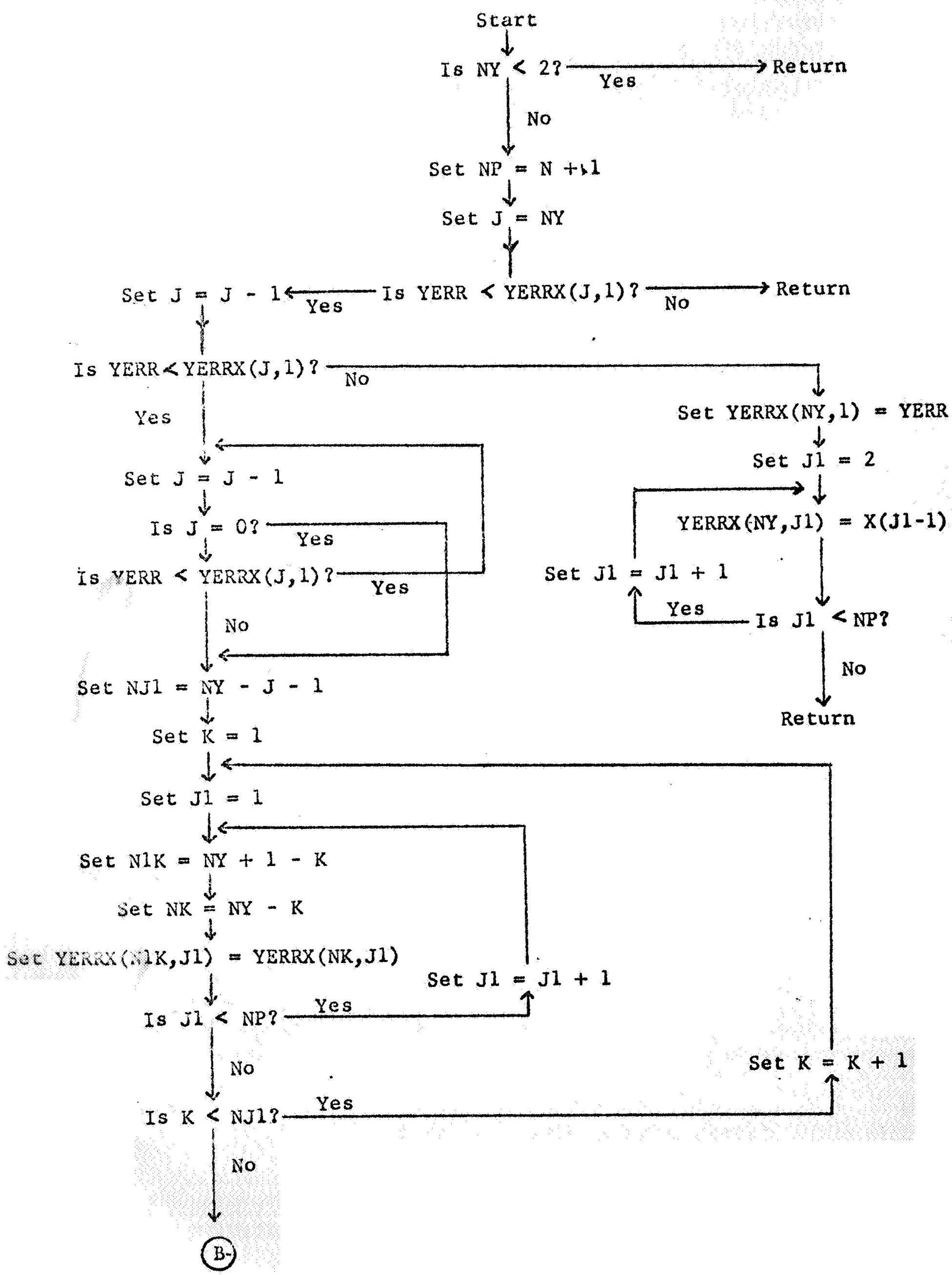


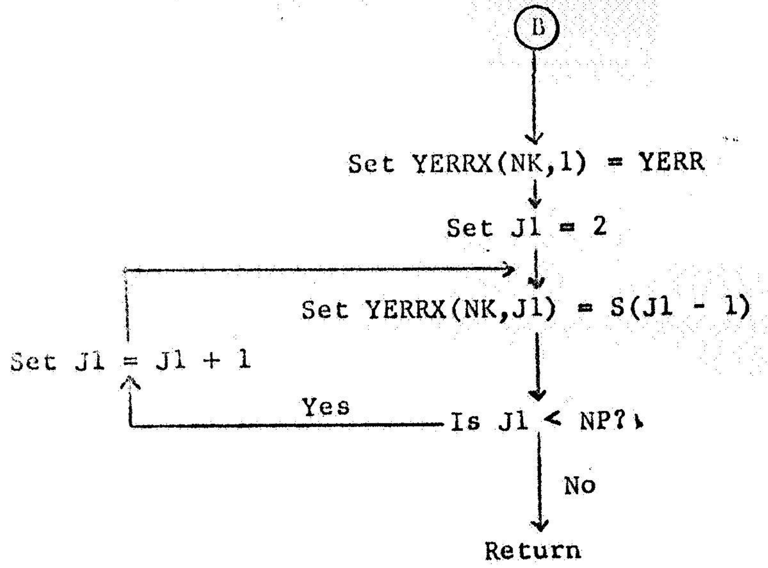
Flow Chart for Subroutine OPT1 (continued)

```
               SUBROUTINE OPTI
         C     GRID SEARCH OPTIMIZATION SUBROUTINE
         C     FOR CDC 6400, HUELSMAN, JUNE 196
000002         COMMON /OPT/X(20),XL(20),XU(20),KX(20),XP(20),Q(20),Z(20),Z(20)
              1,G(20),PARAM(10,7),NOPT(10,10),N,NN,YERR,ITER,EPSTE,ITMAX,ALFA(8)
000002         DIMENSION XM(20,20)
000002         DIMENSION YERRX(10,21)
000002         DATA YERRX/210*0.0/
000002         PRINT 110
000006     110 FORMAT (///1X*OPTI GRID SEARCH SUBROUTINE HAS BEEN CALLED*/)
000006         IF(PARAM(1,1).EQ.0.) PARAM(1,1)=5.
000010         PRINT 120, PARAM(1,1), (NOPT(1,I), I=1,2)
000025     120 FORMAT (
              11X*PARAM(1,1)-NUMBER OF BEST VALUES TO BE PRINTED.*E10.3/
              21X*NOPT(1,1)-PRINT MANY VALUES(1=YES,0=NO,2=ALL)..*I2/
              31X*NOPT(1,2)-MAKE LOCAL SEARCH(1=YES,0=NO)........*I2/)
000025       5 ITER=0
000026         NC=1
000027         DO 30 I=1,N
000031      30 NC=NC*KX(I)
000037       6 DO 10 I=1,N
000041         KKP=KX(I)
000043         XKP=KX(I)-1
000046         XDIF=(XU(I)-XL(I))/XKP
000052         DO 10 J=1,KKP
000054         XJ=J-1
000056      10 XM(I,J)=XJ*XDIF + XL(I)
000071         NY=PARAM(1,1)
000073         DO 35 J=1,NC
000074         ITER=ITER+1
000076         CALL COMB (J,XM,KX,N,X)
000101         CALL ANLYZ
000102         CALL ERR
000103         IF(ITER.GT.1) GO TO 310
000107         DO 300  I=1,N
000110     300 XP(I)=X(I)
000115         YERRP=YERR
000117         DO 305  I=1,NY
000120     305 YERRX(I,1)=1.E+99
000125     310 IF(NOPT(1,1)-1) 15,312,315
000130     312 CALL ORDER (NY,YERRX)
000132         GO TO 34
000133     315 PRINT 14,ITER,YERR
000143      14 FORMAT(1H0*ITERATION*I4,5X*ERROR=*E12.5)
000143         PRINT 16,(X(I),I=1,N)
000156      16 FORMAT (1X*X(I)=*5(E10.3,1X))
000156      15 IF (YERR-YERRP) 20,34,34
000161      20 YERRP=YERR
000163         DO 25 I=1,N
000164      25 XP(I)=X(I)
000171      34 IF(ITER.GE.ITMAX) GO TO 36
000174      35 CONTINUE
000176      36 PRINT 211
000202     211 FORMAT(/1H0*SUMMARY OF BEST POINTS*)
000202         IF(NOPT(1,1).NE.1) GO TO 320
000204         PRINT 215,ITER
000212     215 FORMAT (*0TOTAL ITERATIONS*I4/)
```

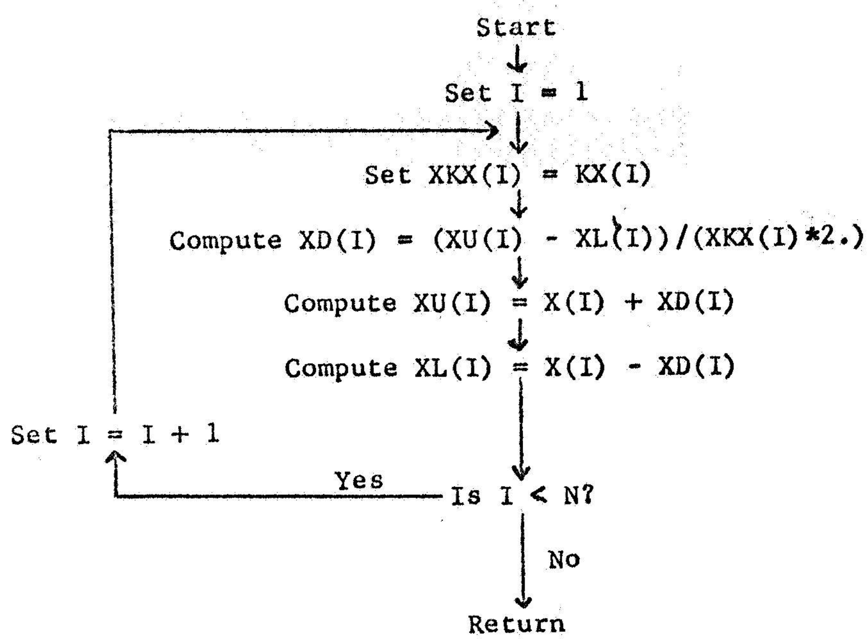
```
      NP2=N+1
      DO 210 I=1,N
      PRINT 212,I,YERRX(I,1)
  212 FORMAT (1H0*NO.*I3,3X*ERROR=*E12.5)
      PRINT 16, (YERRX(I,J1),J1=2,NP2)
  210 CONTINUE
      YERR=YERRX(1,1)
      DO 220 I=1,N
      JP=I+1
  220 X(I)=YERRX(1,JP)
      GO TO 335
  320 I=1
      PRINT 212,I,YERRP
      PRINT 16,(XP(I),I=1,N)
  325 YERR=YERRP
      DO 330  I=1,N
  330 X(I)=XP(I)
  335 IF (ITER.GE.ITMAX) RETURN
      IF (YERR.LE.ERMIN) RETURN
      IF (NOPT(1,2).EQ.0) RETURN
      CALL REDUC
      GO TO 6
      END
```

Flow Chart for Subroutine ORDER

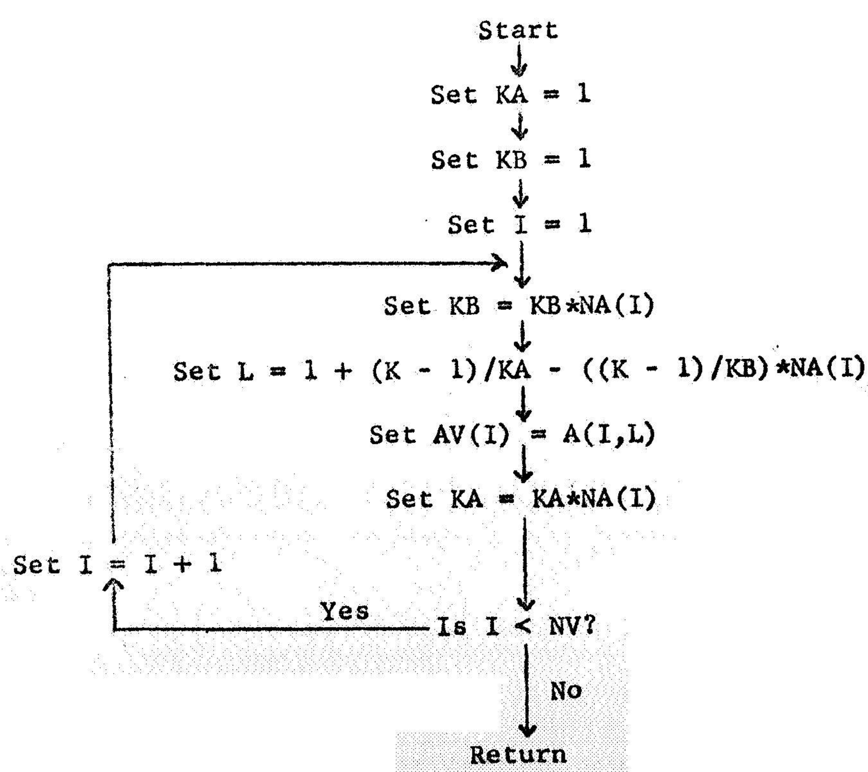



B
Set YERRX(NK,1) = YERR
Set J1 = 2
Set YERRX(NK,J1) = S(J1 - 1)
Set J1 = J1 + 1
Yes
Is J1 < NP?
No
Return

Flow Chart for Subroutine REDUC



Flow Chart for Subroutine COMB



```
                SUBROUTINE ORDER (YERR, NY, YERRX, X)
000007          COMMON /OPT/X(20),XL(20),XU(20),KX(20),XP(20),H(20),R(20),W(20)
               1,G(20),PARAM(10,7),NOPT(10,10),N,NH,YERR,ITER,ERMIN,ITMAX,ALFA(8)
000007          DIMENSION YERRX(10,21)
000007          IF (NY.LT.2) RETURN
  0012          NP=N+1
000014          J=NY
000015          IF(YERR-YERRX(J,1)) 105,105,125
000020      105 J=J-1
000022          IF (YERR-YERRX(J,1)) 305,305,310
000025      305 J=J-1
000027          IF(J) 110,110,115
000030      115 IF (YERR-YERRX(J,1)) 305,305,110
000033      110 NJ1= NY-J-1
000036          DO 120 K=1,NJ1
000037          DO 120 J1=1,NP
000040          N1K=NY+1-K
000043          NK=NY-K
000044      120 YERRX(N1K,J1)=YERRX(NK,J1)
000055          YERRX(NK,1)=YERR
000057          DO 520 J1=2,NP
000060      520 YERRX(NK,J1)=X(J1-1)
000070          GO TO 125
000070      310 YERRX(NY,1)=YERR
000072          DO 315 J1=2,NP
000074      315 YERRX(NY,J1)=X(J1-1)
000104      125 CONTINUE
000104      130 RETURN
000105          END
```

```
                SUBROUTINE COMB (K,A,NA,NV,AV)
000010          DIMENSION A(20,20), NA(20), AV(20)
000010          KA=1
000011          KB=1
000012          DO 110 I=1,NV
  0013          KB=KB*NA(I)
000016          L=1+(K-1)/KA-((K-1)/KB)*NA(I)
000030          AV(I)=A(I,L)
000034      110 KA=KA*NA(I)
000041          RETURN
000041          END
```

```
                SUBROUTINE REDUC
000002          COMMON /OPT/X(20),XL(20),XU(20),KX(20),XP(20),H(20),R(20),W(20)
               1,G(20),PARAM(10,7),NOPT(10,10),N,NH,YERR,ITER,ERMIN,ITMAX,ALFA(8
000002          DIMENSION XKX(20),XD(20)
000002          DO 10 I=1,N
 00004          XKX(I)=KX(I)
000007          XD(I)=(XU(I)-XL(I))/(XKX(I)*2.)
000016          XU(I)=X(I)+XD(I)
000022       10 XL(I)=X(I)-XD(I)
000030          RETURN
000031          END
```