

MPAD REPORT CONTROL

COPY

DO NOT REMOVE



NASA

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

MSC INTERNAL NOTE NO. 66-FM-131

November 4, 1966

# AS-503A/504A REQUIREMENTS FOR THE RTCC: GENERALIZED ITERATOR

By William E. Moore

Analytical Mechanics Associates



MISSION PLANNING AND ANALYSIS DIVISION  
MANNED SPACECRAFT CENTER  
HOUSTON, TEXAS

NASA - Manned Spacecraft Center

RELEASE APPROVAL

1. Type of Document

Internal Note

2. Identification

66-FM-131

Page 1 of 1 Pages

TO:

Chief, Mission Planning and Analysis Division

3. FROM:

Division Mission Planning and Analysis  
Branch Mission Analysis  
Section Analytical Mechanics Associates

4. Title or Subject

AS-503A/504A Requirements for the RTCC: Generalized Iterator

Date of Paper

11/4/66

5. Author(s)

William E. Moore

6. Distribution

Number of Copies	Addressees	Special Handling Methods
	See attached memorandum	
20	Paula/FM5 11-14-66	<p><i>Handwritten notes:</i>            Originals of memorandum to            be prepared by            11/29/66            11/29/66            5954            -bell            mst</p>
4	Linbeck/FM5/ 11-16-66	
10	B. Maza/FM2/11/29/66	
3	Dorothy/FM3 12-6-66	
1	Foggatt/FM3 12-6-66	
2	Walker/FM2 12-7-66	
2	Jones/FM5 3-10-67	
3	Wiley FM5 4/14/67	

This is a change to distribution on Release Approval dated,

This is an addition to distribution on Release Approval dated,

7. Signature of Branch Head

*[Handwritten signature]*

Signature of Division Chief

*[Handwritten signature]*

Date

3 NOV 1966

Signature of Appropriate Assistant Director or Program Manager

Date

8. Change or Addition made by

Date

9. Location of Originals:

UNITED STATES GOVERNMENT

# Memorandum

TO : See list attached

DATE: 3 NOV 1966

FROM : FM/Mission Planning and  
Analysis Division

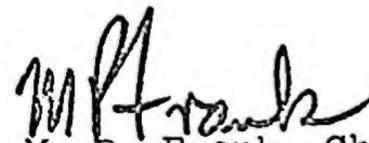
66-FM51-388

SUBJECT: Generalized Iterator Formulation for the RTCC

The enclosed MSC Internal Note No. 66-FM-131 defines formulation for the Generalized Iterator subprocessor. The subprocessor has several applications in both the 503 and 504 systems. The logic as defined is independent of the application; the appropriate "setup" logic must be contained in the supervisor, which calls this subprocessor.

Applications for which this subprocessor will be required include the following:

1. Full mission optimization
  - a. Earth orbit (TLI)
  - b. Translunar coast (MCC)
2. Return-to-earth aborts  
(including TEI)
3. LOI targeting



M. P. Frank, Chief  
Mission Analysis Branch

The Flight Software Branch concurs with the above recommendations.



Lynwood C. Dunseith, Chief  
Flight Software Branch

APPROVED BY:



John P. Mayer, Chief  
Mission Planning and  
Analysis Division

Enclosure



## Addressees:

IBM/R. Hanrahan (20)

FC/J. D. Hodge

/G. S. Lunney

FM/J. P. Mayer

/H. W. Tindall

/C. R. Huss

/M. V. Jenkins

/J. F. Dalby

/J. P. Bryant (5)

/R. P. Parten

/Branch Chiefs

FM6/R. Regelbrugge

FM5/R. Ernull (6)

## cc:

TRW/J. T. Reid (4)

/C. Evaige (2)

Philco/R. D. Harrington

Bellcomm/R. Wagner (5)

EG/Robert Duncan (3)

GV5/M. Rahman

PA1/W. Lee

PD4/A. Cohen

PM3/R. Battey

Technical Information Division, Library (2)

FA/C. C. Kraft, Jr.

/S. A. Sjoberg

/R. G. Rose

FC/T. Weichel (2)

FL/R. L. Thompson

FS/H. E. Clements

FM3/T. Carter (10)

FM5/Mission Analysis Branch (8)

FM5:MPFrank:bc

UNCLASSIFIED

MSC INTERNAL NOTE NO. 66-FM-131

---

PROJECT APOLLO

AS-503A/504A REQUIREMENTS FOR THE RTCC:  
GENERALIZED ITERATOR

By William E. Moore  
Analytical Mechanics Associates

---

November 4, 1966

MISSION PLANNING AND ANALYSIS DIVISION  
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION  
MANNED SPACECRAFT CENTER  
HOUSTON, TEXAS

Approved: 

M. P. Frank III, Chief  
Mission Analysis Branch

Approved: 

John P. Mayer, Chief  
Mission Planning and Analysis Division

UNCLASSIFIED

CONTENTS

Section	Page
1. SUMMARY . . . . .	1
2. INTRODUCTION . . . . .	1
3. FUNCTIONAL FLOW . . . . .	2
4. FUNCTIONAL FLOW CHART . . . . .	4
5. DETAILED FLOW . . . . .	8
5.1 Input . . . . .	8
5.2 Initialization . . . . .	8
5.3 Residual computation . . . . .	9
5.4 Iteration loop . . . . .	10
5.5 Barrier checking . . . . .	12
5.6 Partial derivative matrix calculation . . . . .	12
5.7 Matrix multiplication subroutine . . . . .	13
5.8 Simultaneous equation solves . . . . .	13
5.9 $\lambda$ -sizing subroutine . . . . .	13
6. DETAILED FLOW CHART . . . . .	15

# AS-503A/504A REQUIREMENTS FOR THE RTCC:

## GENERALIZED ITERATOR

By William E. Moore, AMA

### 1. SUMMARY

This note describes the program logic for the iteration scheme to be used in the Real Time Computer Complex for AS-503A/504A. It is a general parameter adjustment scheme which finds the values of the control parameters to achieve a given mission or part of a mission and may optimize the parameter if instructed.

### 2. INTRODUCTION

The program for an iterative method which designs the optimum mission that satisfies a given set of constraints is described in this internal note. It is sometimes no small task to find any set of initial conditions and control parameter values satisfying a set of constraints. The iterative method solves both of these problems, first obtaining a mission that satisfies the constraints (select mode) and then adjusting the initial conditions and control values until a function of these variables achieves a minimum or maximum value (optimizing mode).

The method is completely general; it does not depend on the mission, the initial conditions, control parameters, constraints, or variable to be optimized. The values of the initial conditions and control parameters appear as a vector of independent variables. The values of the constraints and of the variable to be optimized (that is, the variable to be maximized or minimized) appear as a vector of dependent variables. The actual subroutine that computes the dependent variables from the independent variables is referred to as the trajectory computer. The particular trajectory computer to which this iteration scheme applies defines the mission to be obtained or optimized.

A constraint may have either of two forms. It may be an equation, in which a quantity must have a given value, or it may be represented by an inequality. The distinction between these two forms is very important, and is the basis of the success of the method.

The functional flow of the program is described in section 3 and diagramed in section 4. Details of the program are described in section 5 and diagramed in section 6 for those who are interested. Since most of the program is devoted to logic and decisions, there are few complex equations. These are incorporated into the flow chart. The reference contains the mathematical details of the method.

### 3. FUNCTIONAL FLOW

The inputs to the load module include four quantities associated with each independent variable - a switch to designate if the variable is achieved, a first guess (which is to be iteratively updated as the method proceeds), a weighting (scaling) factor, and a step size.

The dependent variables are divided into three classes:

1. Class 1.- Those representing equality constraints
2. Class 2.- Those representing inequality constraints
3. Class 3.- The one representing the variable to be optimized

Five numbers are provided to the load module for each dependent variable. One is a switch to designate if the variable is active, two define the desired values of the constraint for the variable, one designates the class of the variable, and one is a weighting factor which applies only if the variable is of class 2.

To start the computation, the actual weight for each dependent variable and several miscellaneous quantities are computed. The trajectory computer is called, and the first guesses are used as the values of the independent variables. Residuals and the length of the weighted residual vector  $R_L$  are calculated. Class 2 variables which are acceptable are deleted from the weighted residual vector.

The iteration loop begins by establishing a new set of nominal values of the independent variables. If, at any stage, a class 2 variable is close to a boundary, this boundary will act as a barrier. Consequently, control now passes to a barrier-checking procedure, which inspects for this condition, and forces the value of the variable away, if in the select mode, or reassigns the variable to class 1, if in the optimize mode. Next the partial derivatives of all the dependent variables with respect to each independent variable are computed by the secant method.

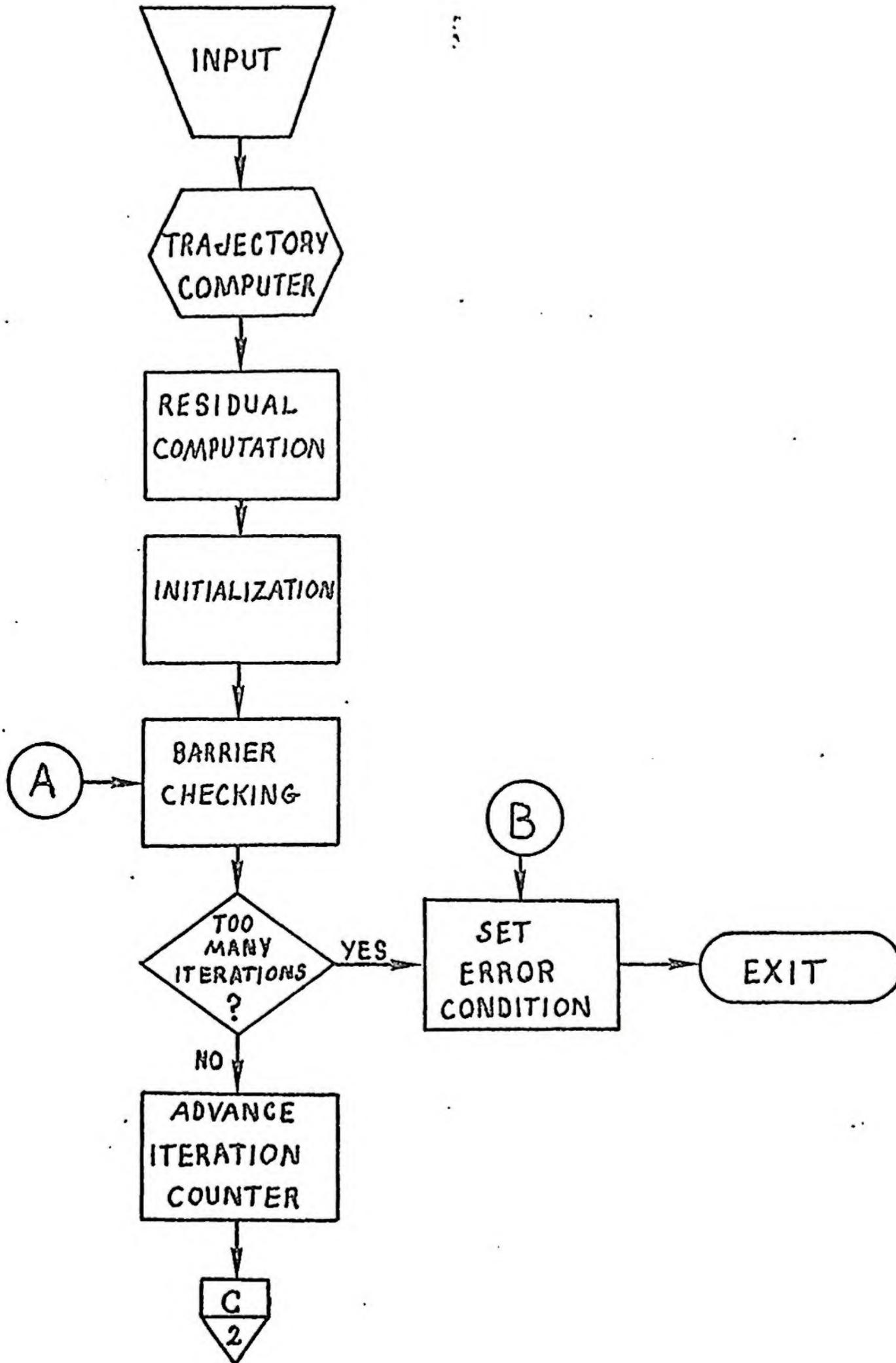
Using these, the weights, the residuals, and the current values of a parameter called  $\lambda$ , increments for the independent variables are computed. These are compared to the step sizes used in the partial derivative calculation. The value of  $\lambda$  is increased and increments are recomputed until the increments are of appropriate size. Next the latest increments are added to the independent variables, and the new values are used in the trajectory computer. The residual vector computer is again called to obtain a new value of  $R_L$ . This is compared with the previous value, and, if it is not smaller, the value of  $\lambda$  is increased and the increments are recomputed until it is.

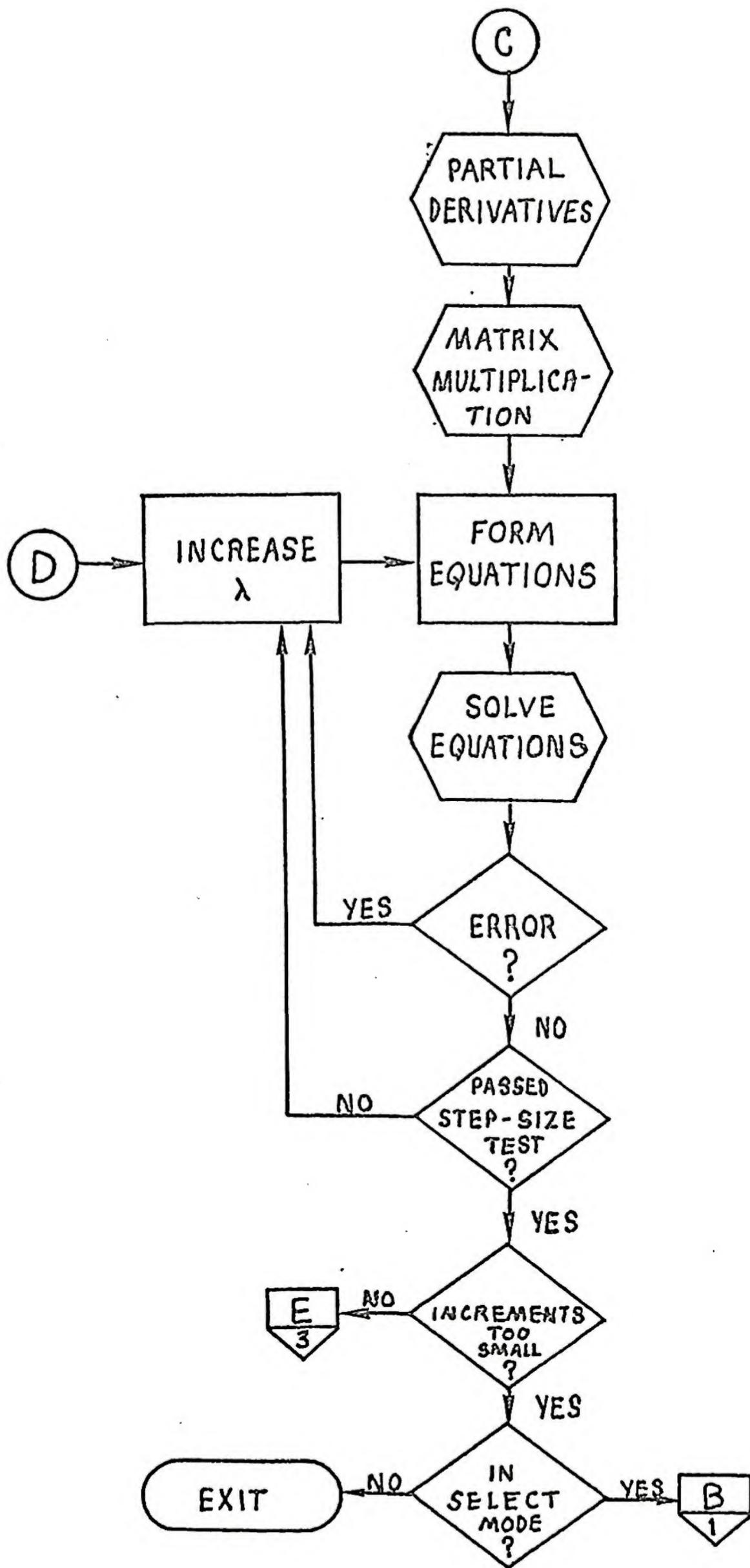
The increases of  $\lambda$  have been of relatively gross size (multiplication by 8). An attempt is made to find a value of  $\lambda$ , one-half or one-quarter as big, which might also be appropriate. Finally, the value of  $\lambda$  is decided, and the values of the independent variables are the ones used in the next nominal trajectory. The iteration counter is advanced,  $\lambda$  is reduced in value, and control passes back to the beginning of the loop.

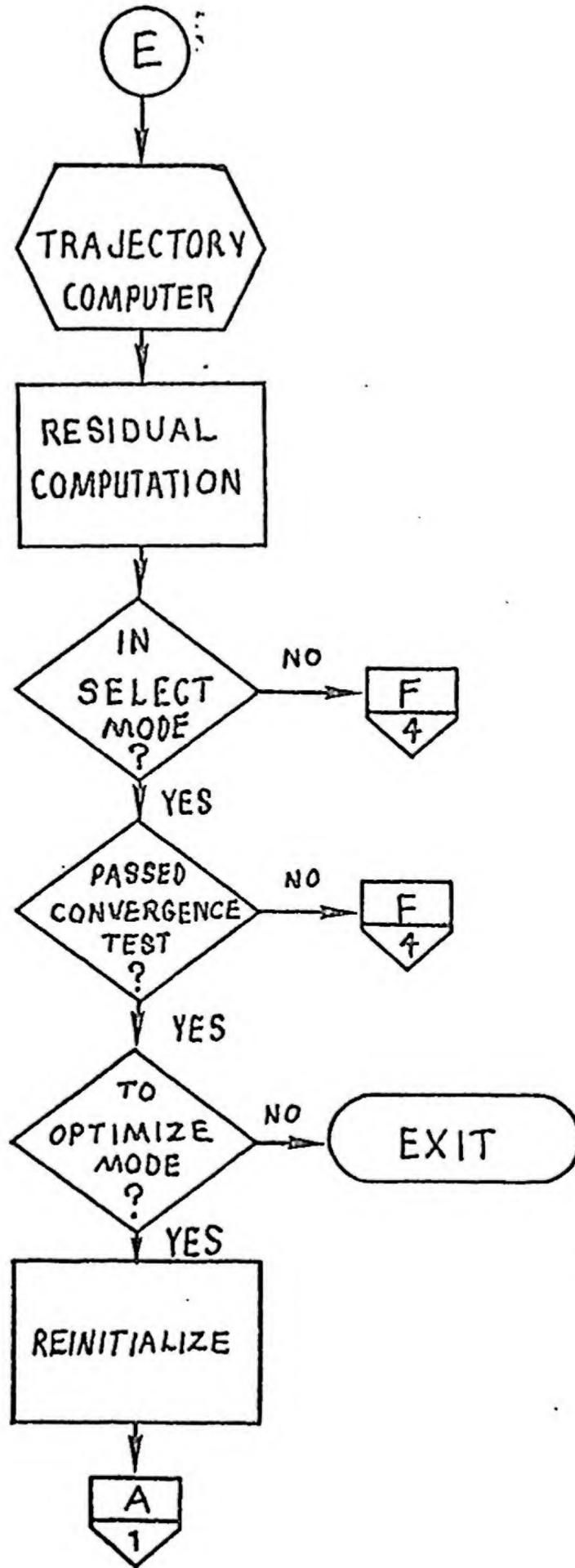
The select mode ends when all of the class 1 and 2 variables satisfy their constraints. Then the variable to be optimized is given a weight and introduced into the problems. The optimizing mode ends when no further change may be made in the values of the independent variables, or when the maximum number of iterations for that mode is exceeded.

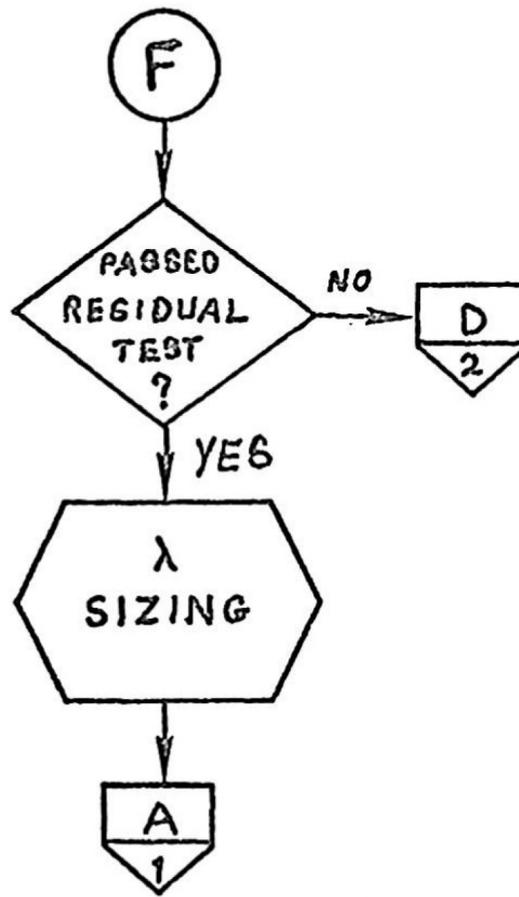
The results are the final values of the independent variables, and their associated dependent variable.

4. FUNCTIONAL FLOW CHART









## 5. DETAILED FLOW

### 5.1 Input

The load module is capable of handling up to twenty candidates for independent and up to twenty candidates for dependent variables. For each of the independent variables, four input quantities are required. These are a switch to show if the variable is active or not, a first guess of the value of the variable, a weight, and a step size to be used in partial-derivative computations. The latter two are needed only when the variable in question is active.

All of the dependent variables can be thought of as having desired values. For consistency, the value for each variable is represented by two numbers, a lower and an upper limit of acceptability. For variables in class 1, these limits are the desired value in the equation decreased and increased, respectively, by a suitably small tolerance. For variables in class 2, several situations can arise. If the inequality is two-sided, the limits are the obvious values from it. If the inequality is one-sided, the dependent variable in question may have a natural limit on the other side which can be used in combination with the given limit. If even this fails, the interval of acceptability must be chosen so that it contains exactly one solution to the problem described by all the other constraints and the variable to be optimized. For a variable in class 3, the notion of acceptability reduces to a convenient fiction, and the limits (usually having the same value) designate the direction in which the variable to be optimized should move. If its value is smaller than the limits, the iterative adjustments will increase it; if its value is larger, the adjustments decrease it.

Thus, each dependent variable will have five input quantities - a switch to show whether the variable is active or not, lower and upper limits, the class designator, and the weighting factors for class 2 variables.

Other inputs are used for housekeeping. The procedure should know how many iterations are to be tried in the select mode and in both modes before terminating. Furthermore, there should be a switch to allow starting in the optimize mode if desired.

### 5.2 Initialization

The switch must be examined to determine how many of each set of variables are active and to establish an array of indices for each set. One of these arrays permits the values of the active independent variables

to be assigned the proper locations for the trajectory computer to use; the other array allows the values of the active dependent variables candidates to be arranged next to each other after the trajectory computer is finished. The two processes involved are called the open-ranks and close-ranks operations, respectively. Each call to the trajectory computer must be preceded by a call to an open-ranks subroutine and followed by a call to a close-ranks subroutine. Hereafter, the term "trajectory computer" will always refer to the set of all three subroutines.

After resetting the iteration counter, the trajectory computer is referenced, using the input first guesses as the values of the independent variables. If this trajectory cannot be computed, the whole procedure is abandoned.

Next certain computations have to be performed once for all iterations. First, average values for each dependent value are calculated half-way between the limits. The weight for each class 1 variable is obtained from the deviations of the limits from the average value as follows. Let  $\bar{y} - \delta_j$  and  $\bar{y} + \delta_j$  be the lower and upper limits, respectively. Then  $\frac{2^{-40}}{\delta_j^2}$  are the weights  $w_j$ . These weights are averaged

using a crude geometric mean to get a number  $w_{avg}$  to be used for the class 2 variables. Each class 2 variable has a weight equal to its weight factor times  $w_{avg}$ . The class 3 variable is assigned a weight based on the value resulting from the trajectory computer. If there are  $k_1$  dependent variables of class 1 and if  $y - \bar{y}$  is the residual of the variable to be optimized, the weight is  $(10^{-4}) (2^{-40}) k_1 / (y - \bar{y})^2$  if starting in the select mode, or  $(2^{-32}) f k_1 / (y - \bar{y})^2$  if starting in the optimize mode. The factor  $f$  is the weight factor associated with the class 3 variable.

Another calculation pertaining to class 2 variables obtains 0.2% of the widths of their intervals. These numbers are used during the iterations to test how near a variable is to one of its limits. Finally, the parameter,  $\lambda$ , is set to a small value,  $2^{-28}$ .

### 5.3 Residual Computation

The residual vector, weighted residual vector, and its length are now computed in the residual package. The residual vector is the set of differences between each dependent variable's value  $y_j$  and its

average value  $\bar{y}_j$  as derived above. Now the class designators  $c_j$  have been assigned the value 1 for class 1 and 3 variables and 0 for class 2 variables. Thus, the weights  $w_j$  are replaced by  $c_j w_j = w_j$  for those variables within the given limits; for variables outside the limits, the weights  $w_j = w_j$  are retained. The weighted residual vector is  $w_j (y_j - \bar{y}_j)$ , and its length  $R_L = \sum_{j=1}^n w_j (y_j - \bar{y}_j)^2$  where  $n$  is the

total number of dependent variables.

If, while checking the values of the dependent variables, it should be found that all except the variable to be optimized are within their intervals, and if the iterations are still in the select mode, that mode is terminated. If there is to be an optimize mode, the weight on the variable to be optimized is reinitialized, and the iterations proceed in the optimize mode. If there is no variable to be optimized, the procedure is finished.

The initial trajectory is the first in a sequence of trajectories designated a "lost good trajectory". As the iterations are performed, a new good trajectory is one that is better than the last good trajectory, in the sense that its value for  $R_L$  is smaller than that of the old one.

#### 5.4 Iteration Loop

Each trajectory is defined by the values of the independent variables used to get it. Thus the values of the independent variables used to obtain the last good trajectory of a previous iteration must be saved separately from the current values of the independent variables, and are the nominal values of the independent variables. This, then, is the first step of each iteration, the transferral of the last good values into the nominal position.

There follows a lengthy procedure of barrier checking, in which each class 2 variable is examined to see where its value is in relation to its acceptable interval, and appropriate action is taken to treat the intervals. A discussion of this portion of the load module is postponed until a later section.

At this point, the iteration counter is advanced, and a test is made to see if all the allowable iterations have been used in either mode.

Next, the partial derivative matrix subroutine is called, followed by a call to a specialized matrix multiplication subroutine, which processes the partial derivative matrix  $P$ , the dependent variable weights  $w_j$ , and the weighted residual vector to get  $P^T W P$  and  $P^T W \Delta y$ , where  $w$  is the matrix having the weights  $w_j$  along its diagonal and  $\Delta y$  is the residual vector (so that  $w \Delta y$  is the previously computed weighted residual vector).

Now, using the current value of  $\lambda$  and the independent variable weights, and a vector of adjustments  $\Delta x_i$  to the independent variables is obtained by referring to a simultaneous equation solver. If no solution can be obtained by the equation solver,  $\lambda$  is multiplied by 8, and the vector of adjustments is recomputed.

If there is a solution, the absolute value of each component is tested against 65536 times the corresponding step-size. To guard against wild excursions of the independent variables, if any component is too big,  $\lambda$  is multiplied by 8 and the vector of adjustments is recomputed. In this way, the value of  $\lambda$  builds up until all components are small enough.

Now the independent variables are incremented by their adjustments and a new trajectory computation is done. The resulting dependent variables are used in the residual computation. If the new value of  $R_L$  is not smaller than the old value, then  $\lambda$  is multiplied by 8, and new increments in the independent variables are computed as above. This procedure continues until the new value of  $R_L$  finally is smaller than the old value.

Meanwhile, each time the independent variables are incremented, and the resulting values are compared with the original values. If no change is detected, the increments have been choked to negligible values. When this happens in the select mode, it indicates that no solution to the problem as presented exists anywhere in the neighborhood of the current values of the independent variables, which constitute the best choice of these values in the least squares sense. When this happens in the optimize mode, the optimum value of the variable to be optimized has been obtained, and the current values of the independent variables are the answer to the original problem.

The  $\lambda$ -sizing subroutine is now called to further refine the value of  $\lambda$  in an attempt to take larger corrections to the independent variables. The last good trajectory from this subroutine becomes the new nominal trajectory and control returns to the beginning of the iteration loop.

### 5.5 Barrier Checking

This procedure is carried out only for the nominal trajectory available at the beginning of each iteration. It pertains only to the class 2 dependent variables, which will, for the rest of this section, be called "variables".

In the select mode, search is made for variables which are inside the acceptable interval, but only just inside, that is, they are within 0.2% of full range of one of the limits. When a variable has such a value, a move procedure begins. The limit near the value of the variable is temporarily replaced by its opposite limit, thus shrinking the interval of acceptability to zero length. The residual vector and its length are then recomputed. The move counter, which has been continually reset at every iteration, now begins to count, and control passes into the basic iterations loop. As the iterations proceed, additional variables may have values which would start a move procedure. If this happens, the limits are treated in the same way, and the move counter begins to count from 1 again. Finally when the move counter reaches 6, the limits are restored to their values, as originally input, and the move counter reverts to being reset at every iteration.

In the optimize mode, there are two procedures, a lock procedure, for variables already locked, an unlock procedure. The lock procedure is as follows: If any variable has a value outside the acceptable limits, it may be greater than the upper limit or less than the lower limit. In the former case, the value of the lower limit is replaced by the value of the upper limit; in the latter, the value of the lower limit takes the place of the upper limit. In either case, the variable is treated like a class 1 variable in the residual computations.

Both the number of iterations during which the value of a previously locked variable is inside the input limits and the number of iterations during which the value is outside the limits are known. If any time after the inside exceeds the outside by three or more, the unlock procedure restores the original input limits. When the limits are restored, the next iteration the required value is taken to be 1.6% of full range of the original interval of acceptability. Thereafter, the interval returns to its original form, and the variable is again treated like a class 1 variable in the residual computation.

### 5.6 Partial Derivative Matrix Calculation

Each independent variable value  $x_i$  is changed successively, and the trajectory is differentiated by  $\Delta x_i$ , the corresponding step-size. The trajectory compared, using the new set of independent variables thus

formed. The resulting values of the dependent variables will be denoted by  $y_j(x_i + \Delta x_i)$ . Let  $y_j(x_i)$  be the nominal values of the dependent variables. Then, the elements of the partial derivatives matrix P

$$\text{are } P_{ji} = \frac{\partial y_j}{\partial x_i} = \frac{y_j(x_i + \Delta x_i) - y_j(x_i)}{\Delta x_i}$$

Thus, there are as many calls to the trajectory computer as there are independent variables, each one generating one column of P. If any of the references to the trajectory computer fails, a step in the reverse direction is attempted, and if its trajectory succeeds, the required partial derivative may be obtained from it. If steps in both directions fail, the whole procedure must halt.

### 5.7 Matrix Multiplication Subroutine

The inputs to this routine are a matrix P, which has n rows, a vector of n weights,  $w_j^!$ , and a vector of n weighted residuals  $W\Delta Y$ .

Each row of P is multiplied by its corresponding weight,  $w_j^!$ , giving a matrix WP and vector  $(W\Delta Y)$ , both of which are multiplied  $^j$  by the transpose  $P^T$ . This is done, in the usual way, by multiplying each of the columns of WP and  $W\Delta Y$  by each of the columns of P.

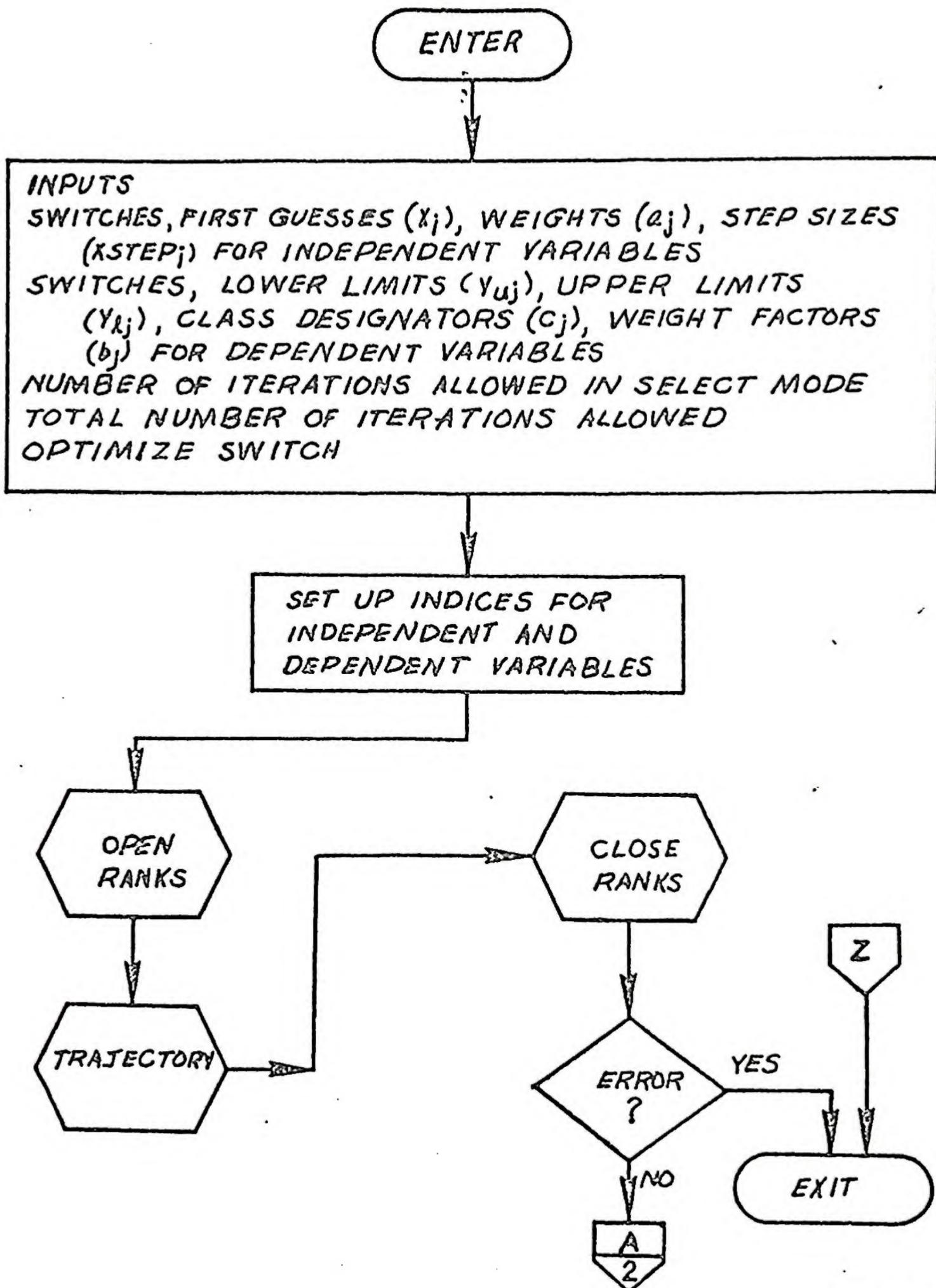
### 5.8 Simultaneous Equation Solves

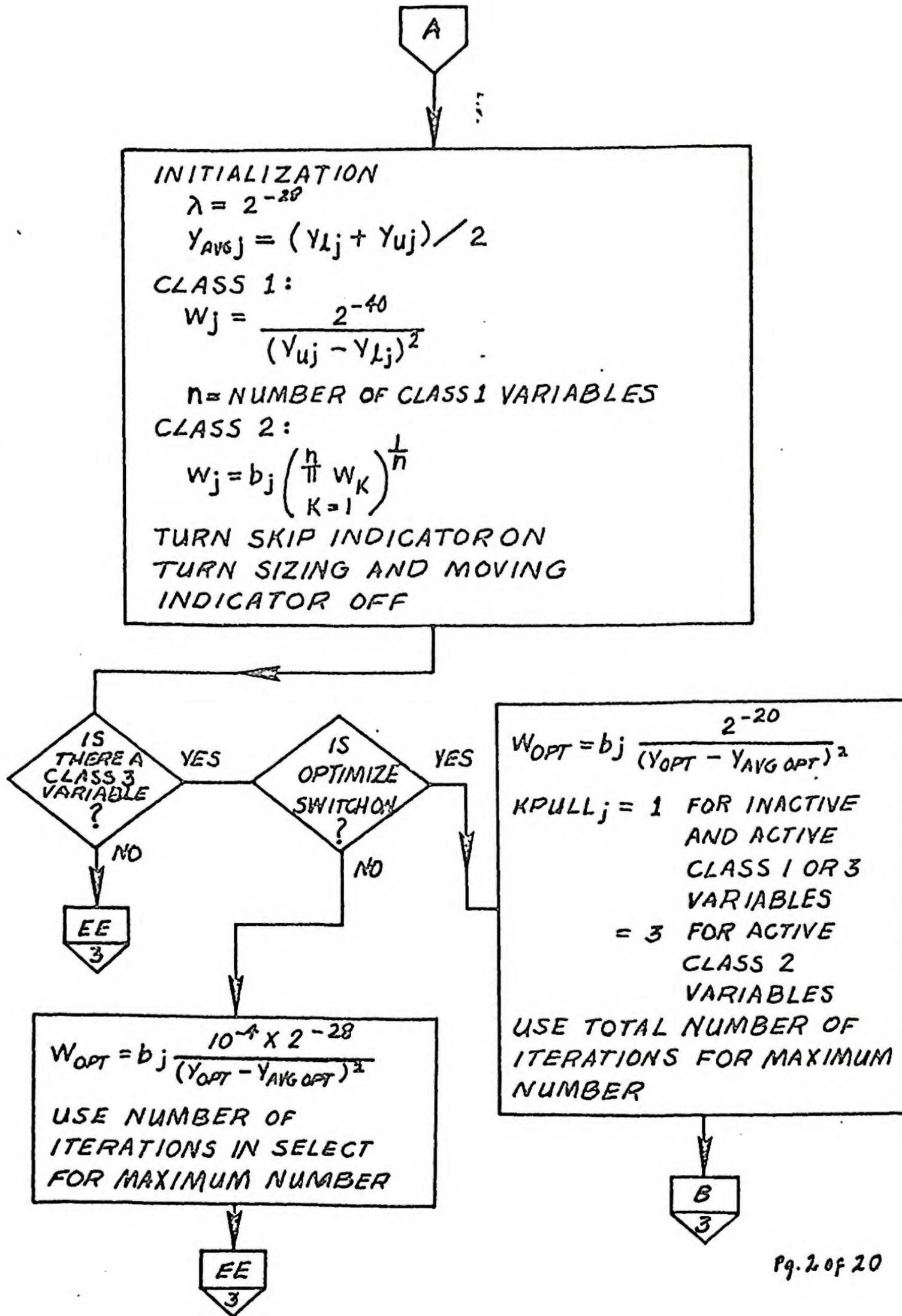
The best method of solving simultaneous equations is to use a straight-forward elimination technique, followed by a back solution. If at any stage, division by a number small enough to cause underflow or division by zero would be required, the remaining equations may be interchanged. If all attempts at this fail, the subroutine returns with an error indication.

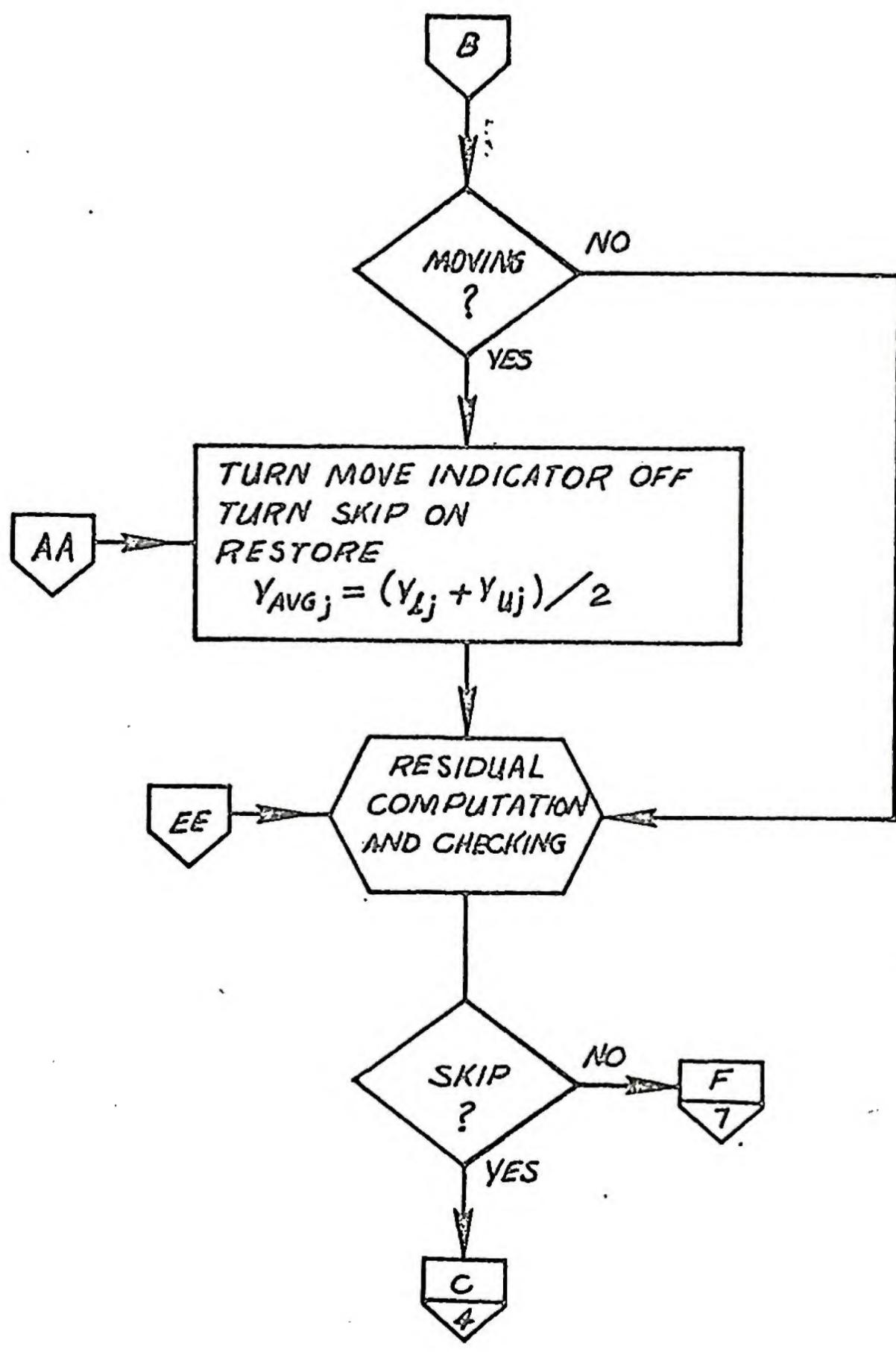
### 5.9 $\lambda$ -sizing Subroutine

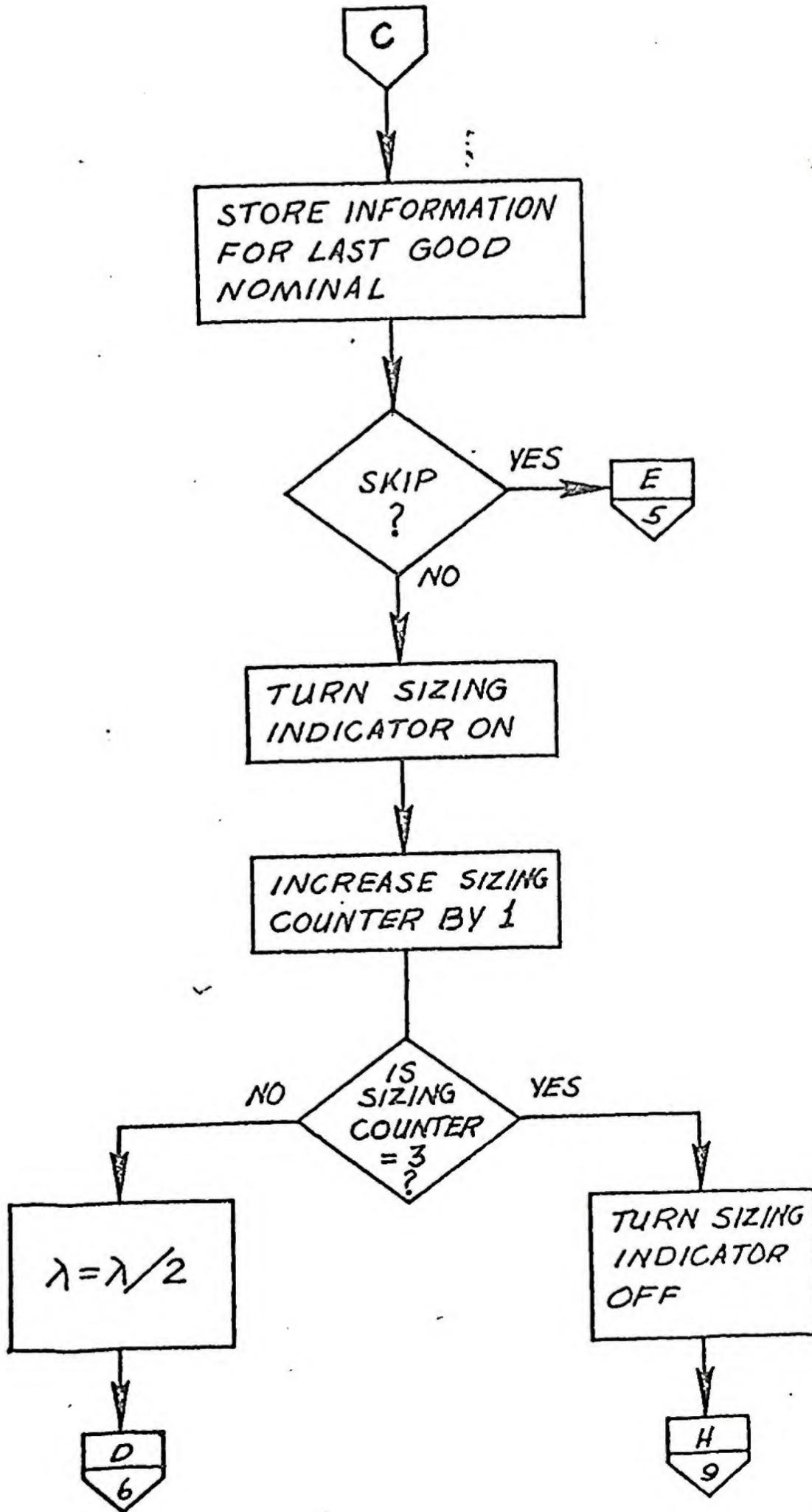
When a value of  $\lambda$  has been arrived at by the above process, it is tested to see if a slightly smaller value with bigger increments might be even better. Thus, the latest values of the independent variables become the values for the last good trajectory, but we continue to work from the values we previously had, still using them as nominal values.  $\lambda$  is replaced by  $\lambda/2$ , and the increments are again computed. These are tested by the same procedures as before, comparison with a multiple of

the step-size and comparison with the value  $R_L$  from the last good trajectory. If they pass the tests, these latest values of the independent variables now define the last good trajectory,  $\lambda$  is replaced by  $\lambda/2$ , and the whole process is repeated once more. If either of the two values of  $\lambda$  fails to produce a better trajectory, the last good trajectory is the result of the subroutine.

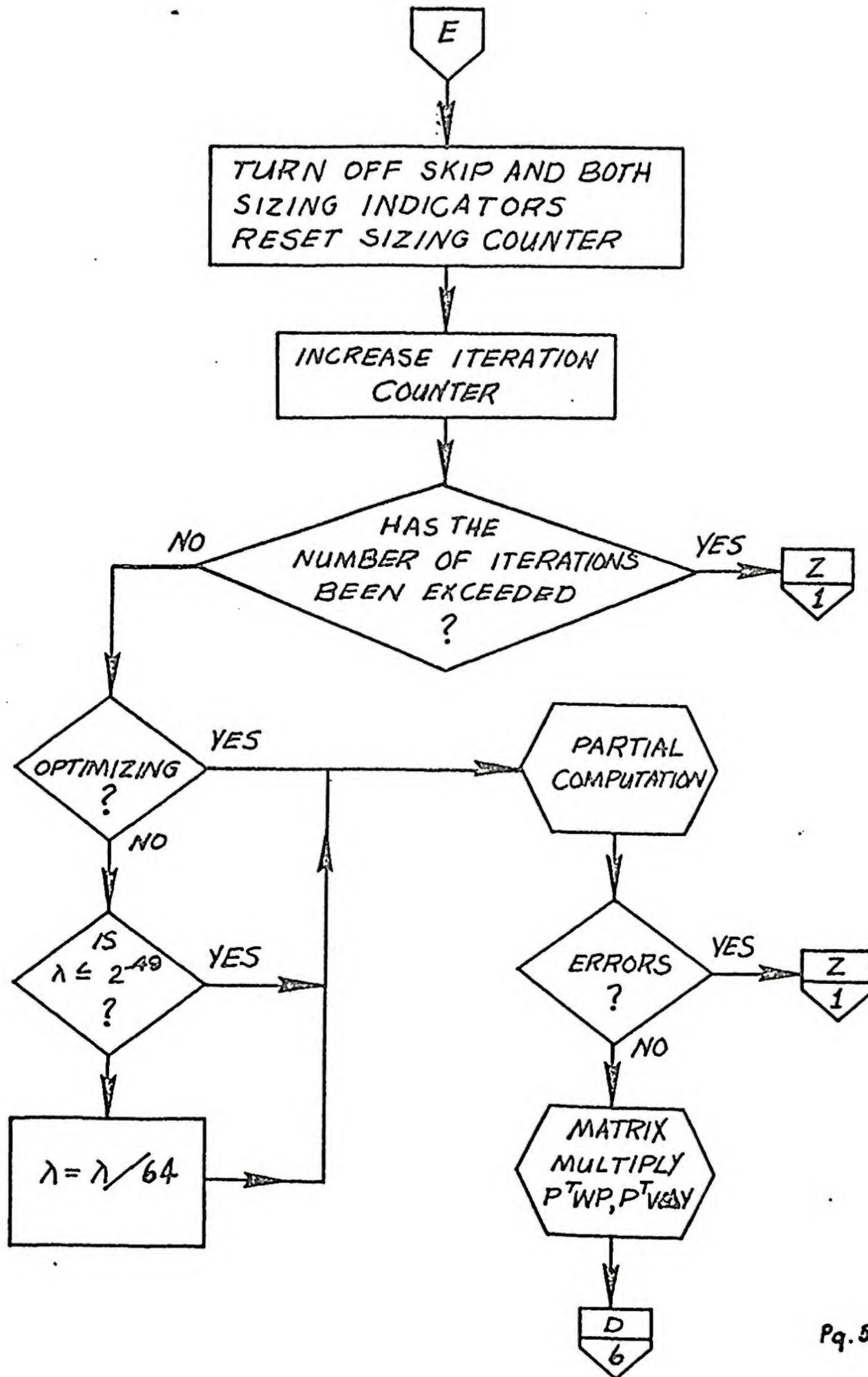


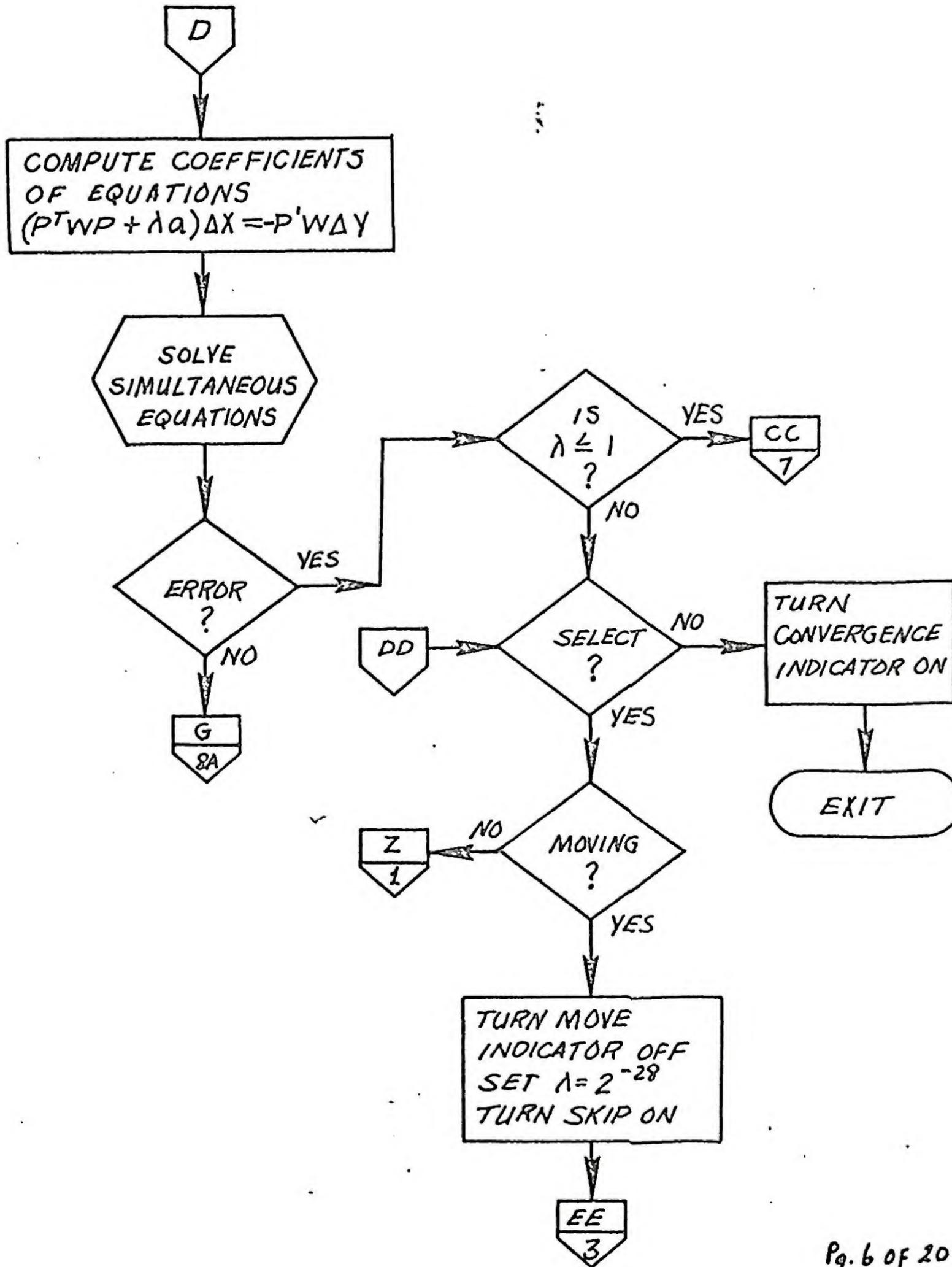


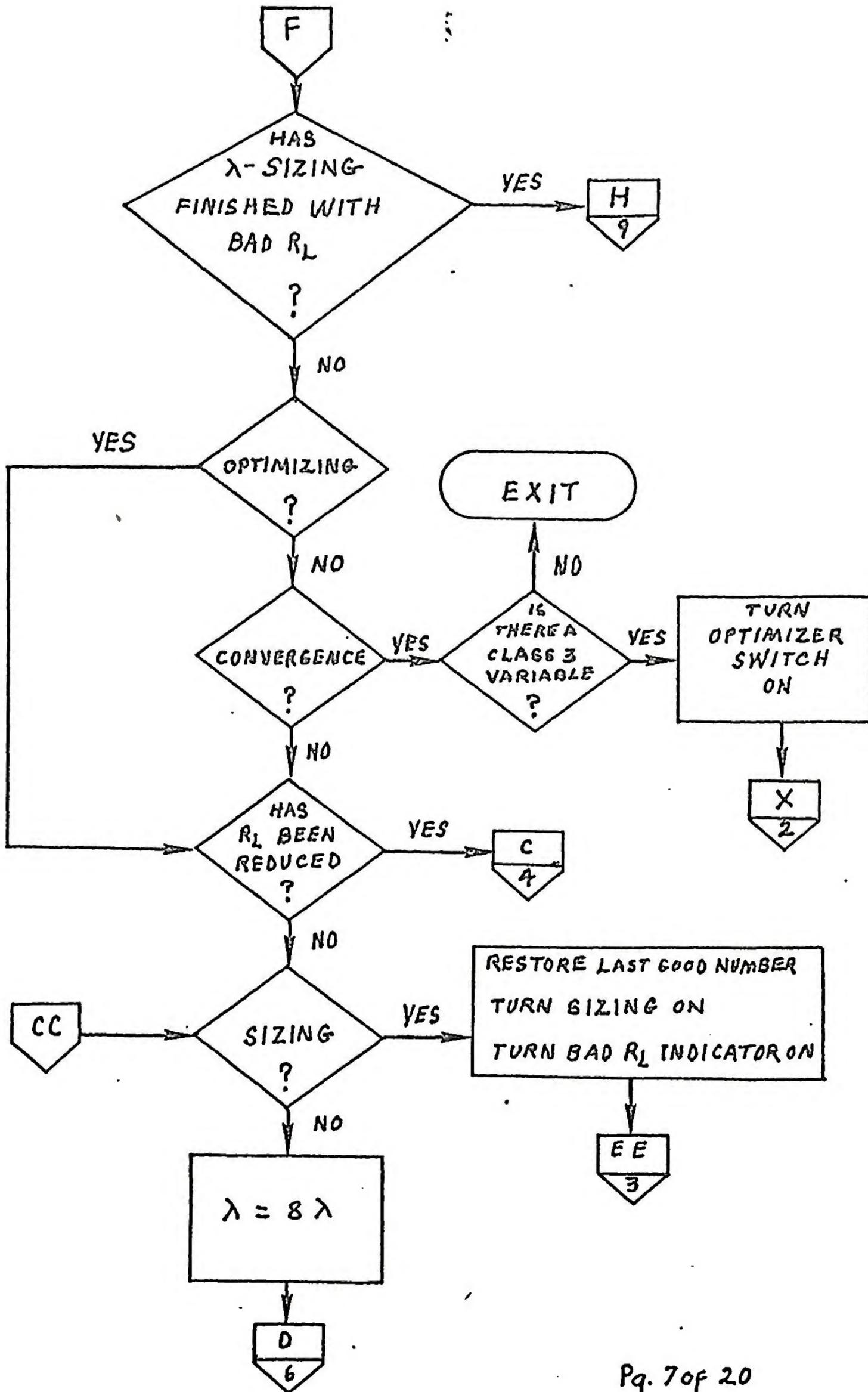


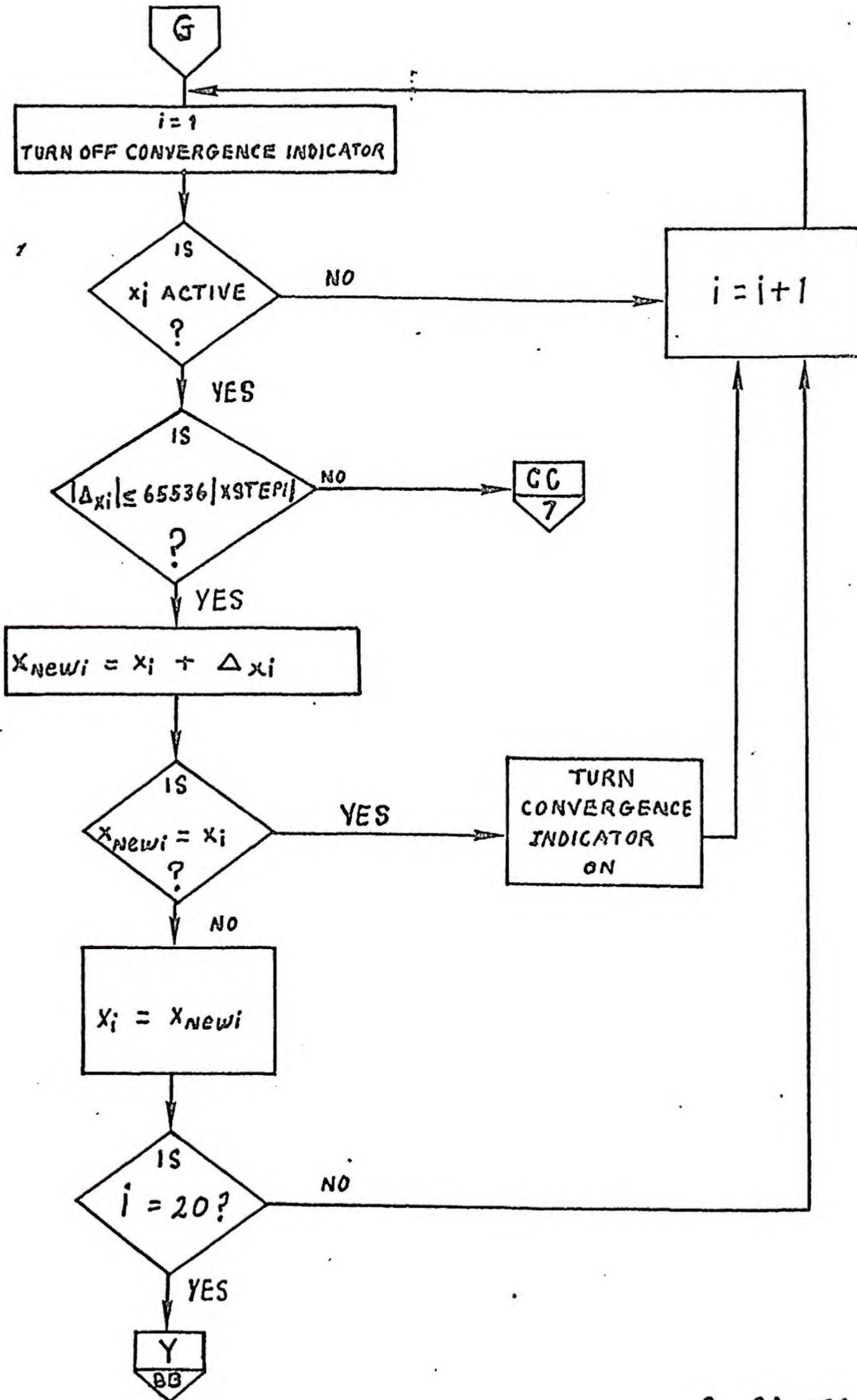


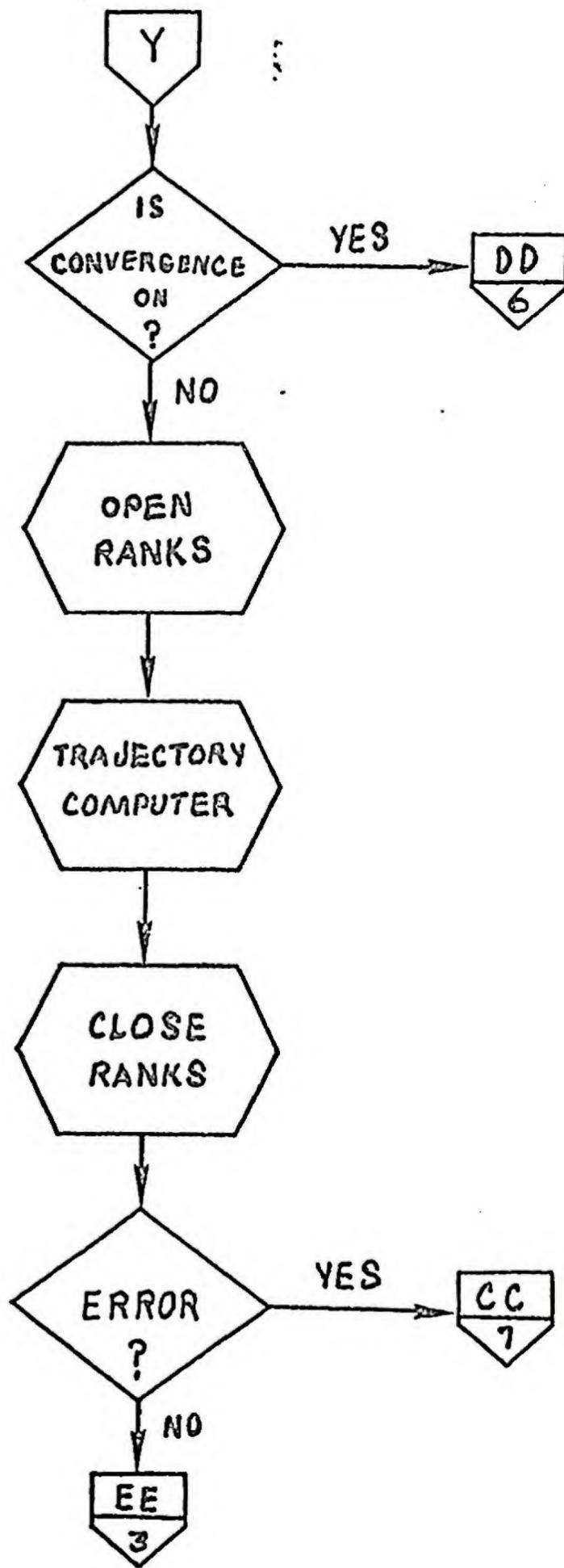
Pg. 4 of 20



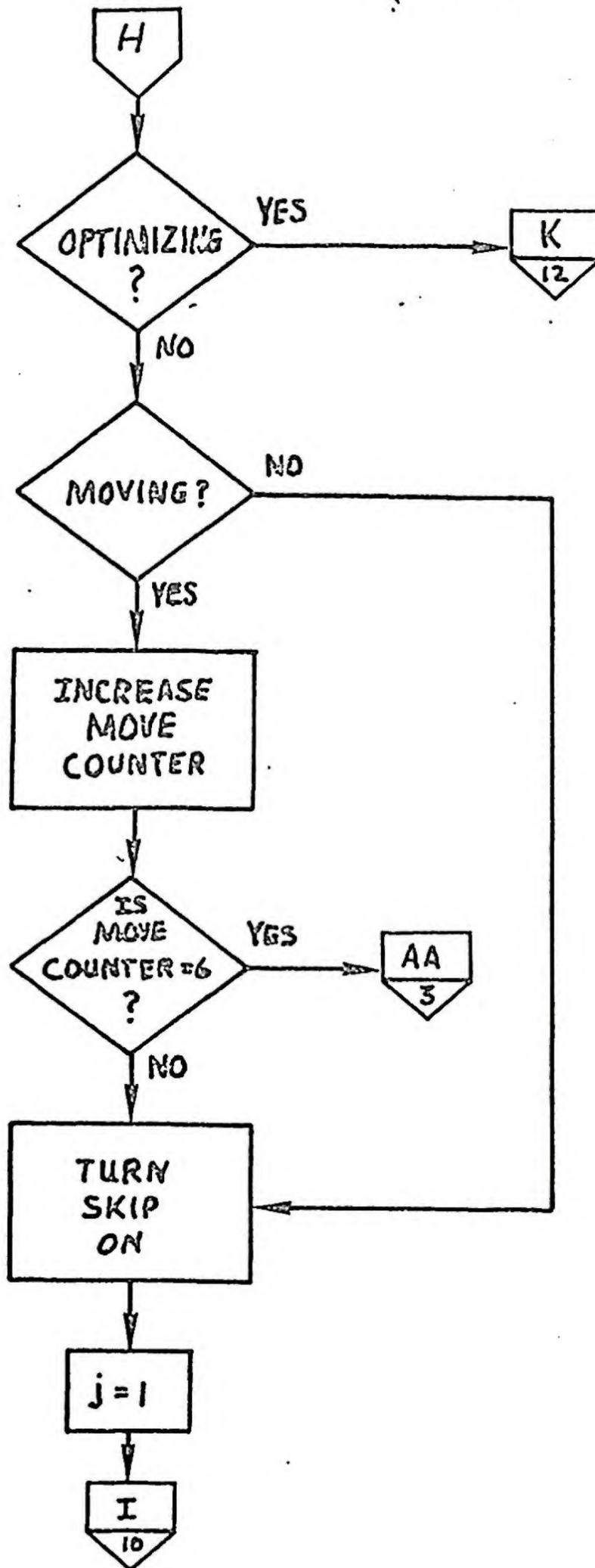


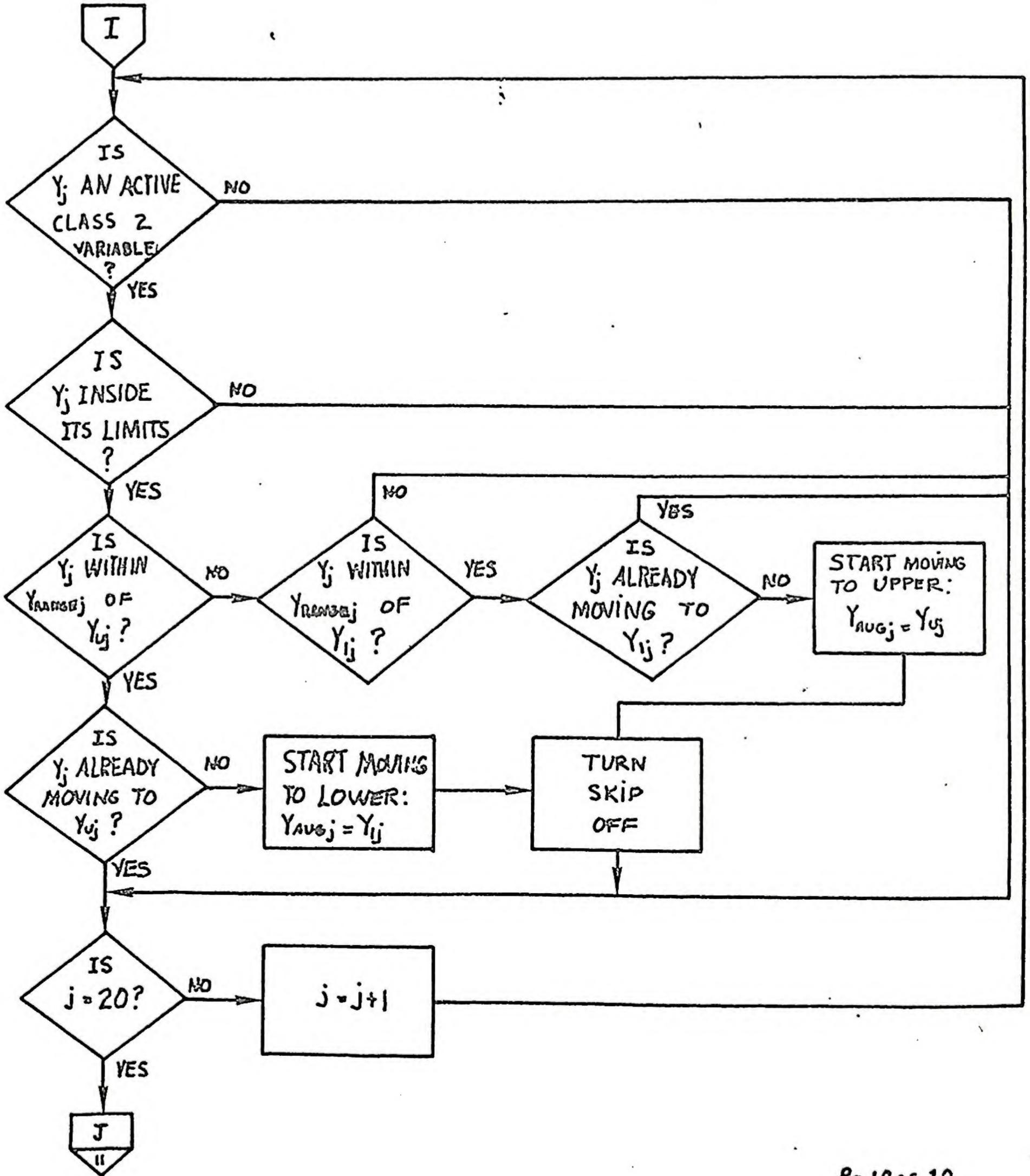


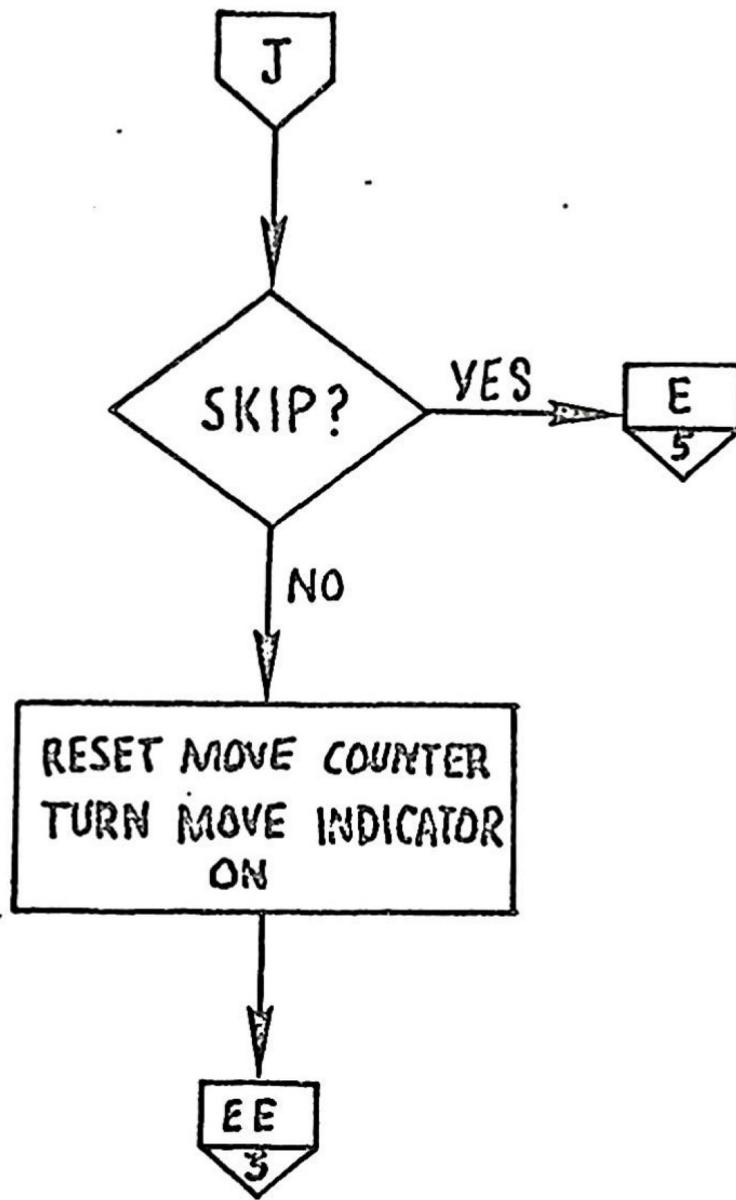




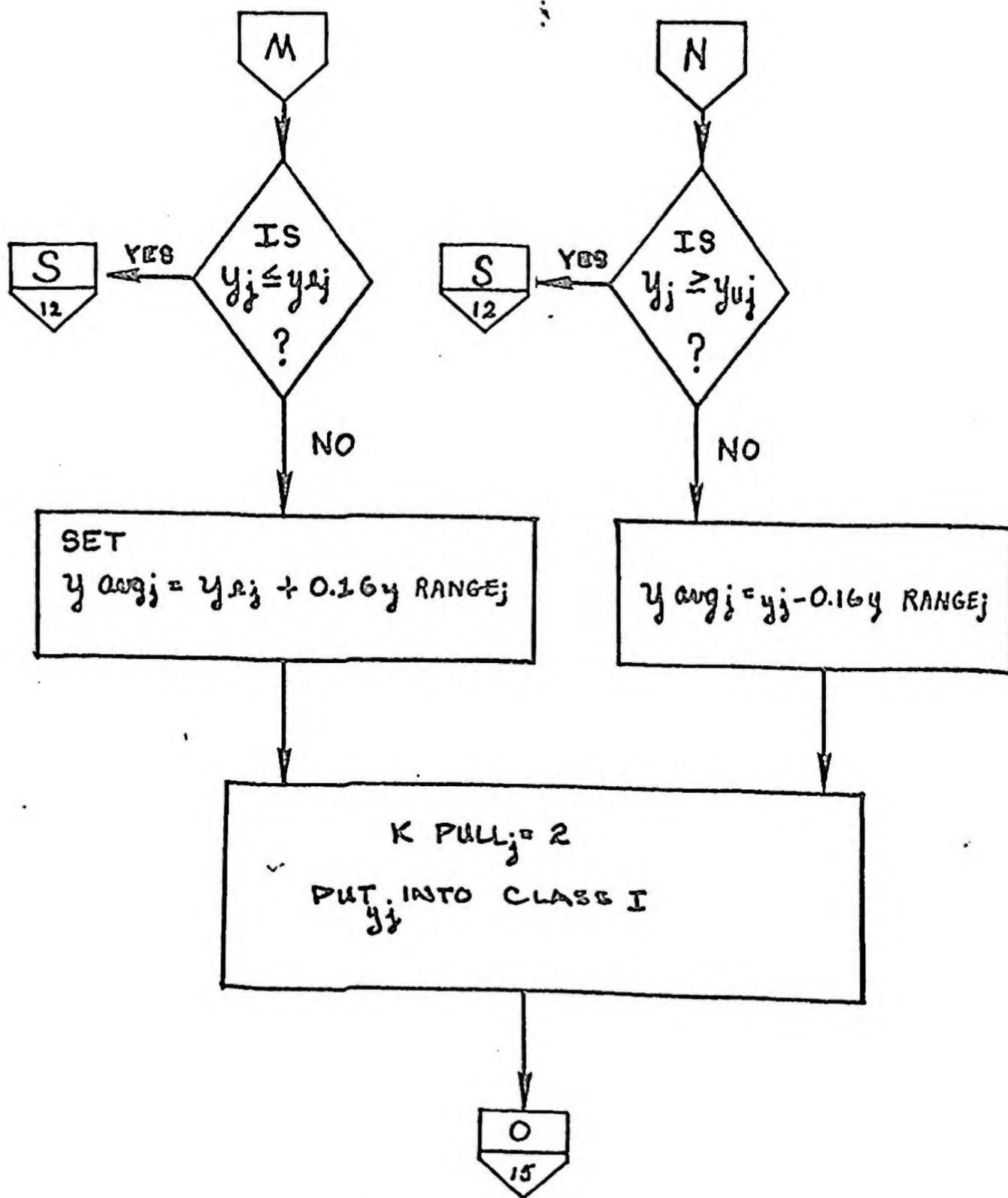
### BARRIER COMPUTATION

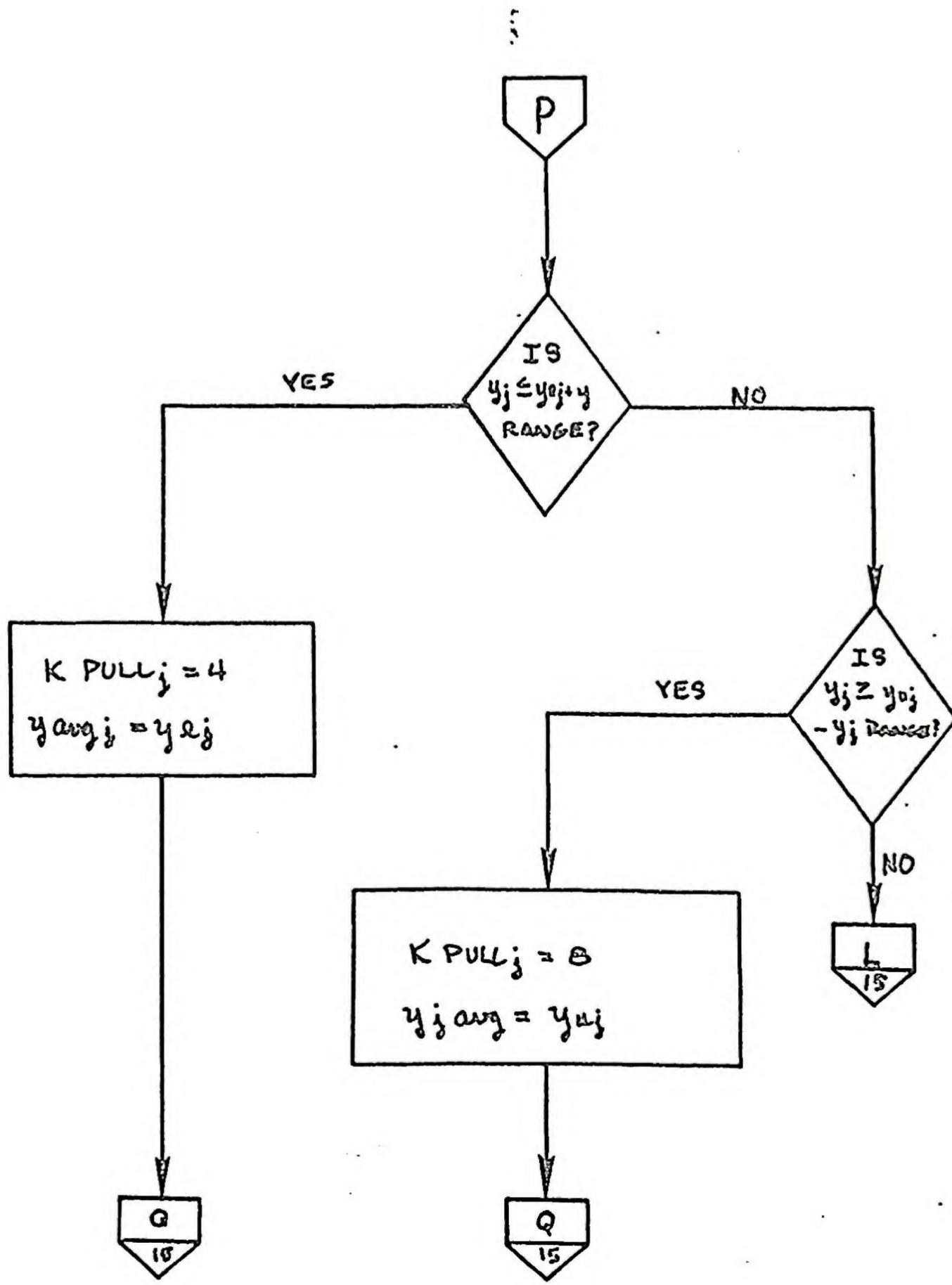


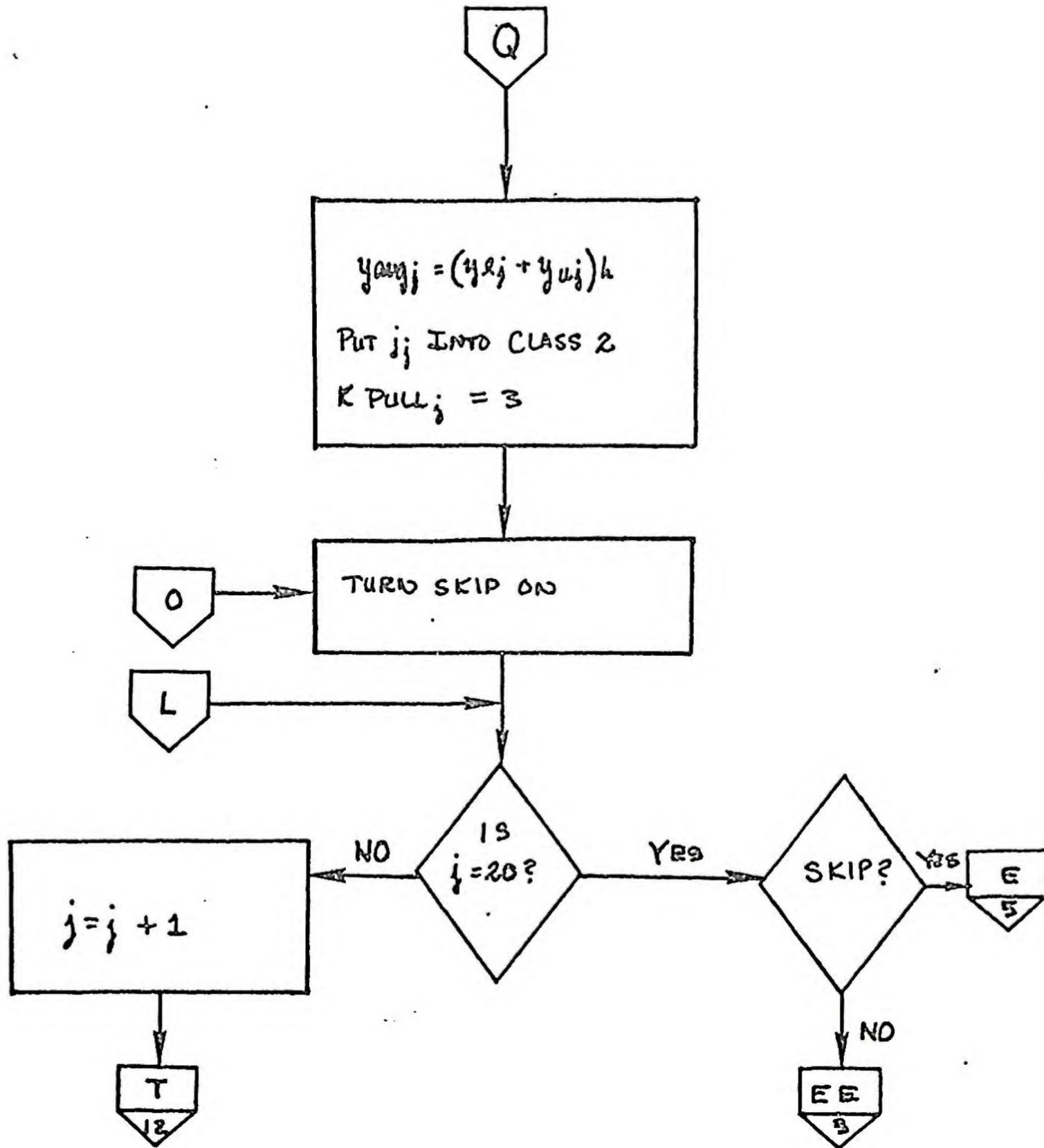




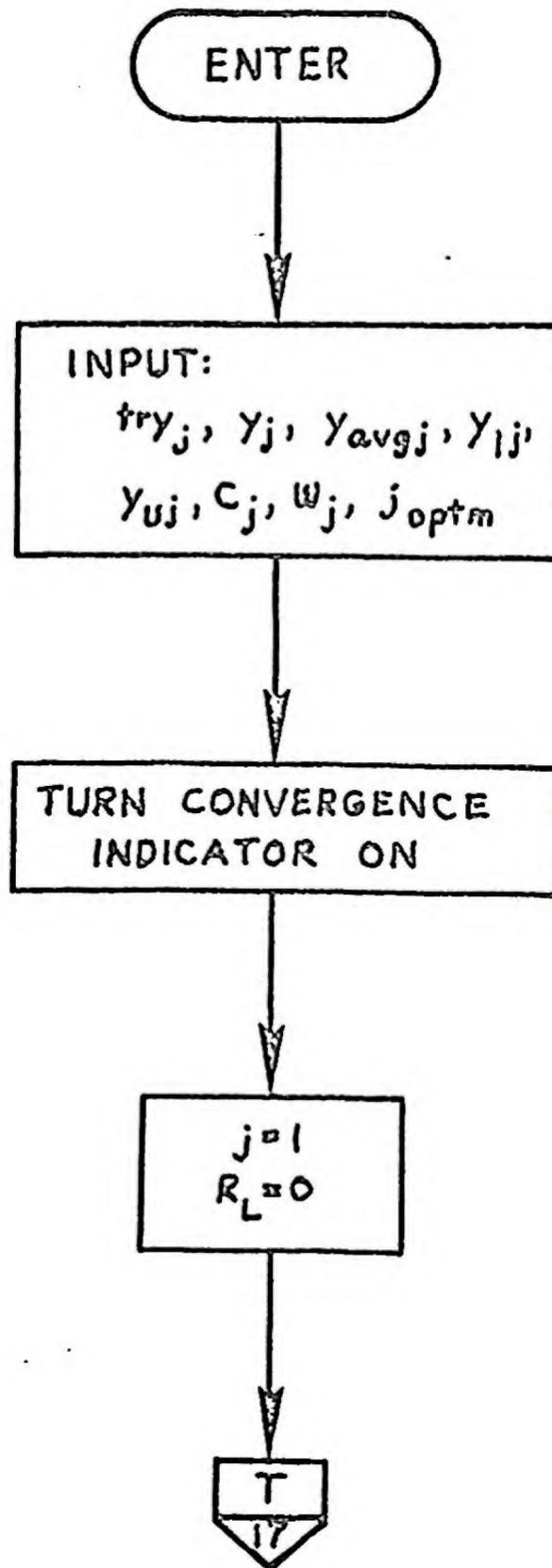


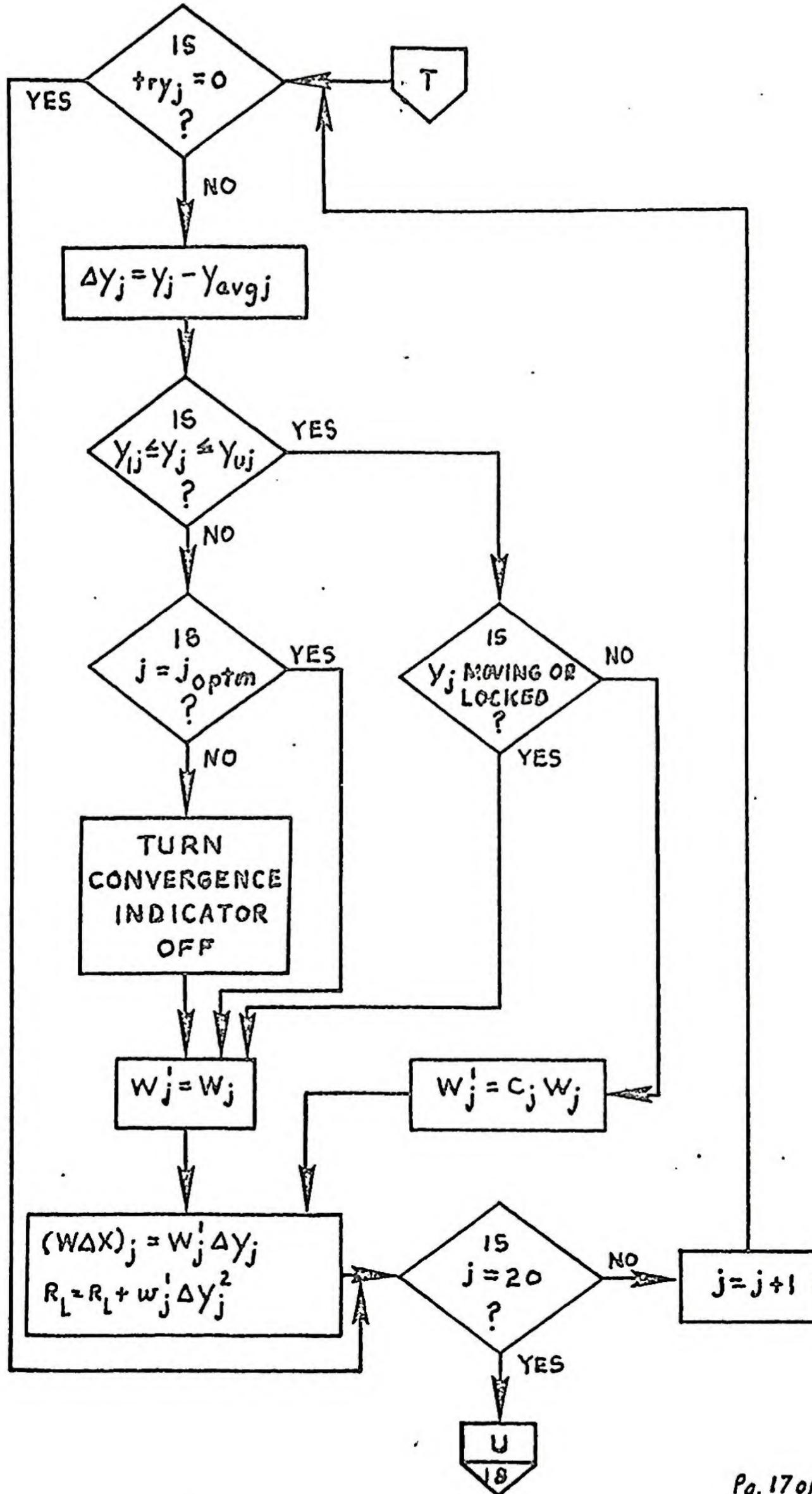


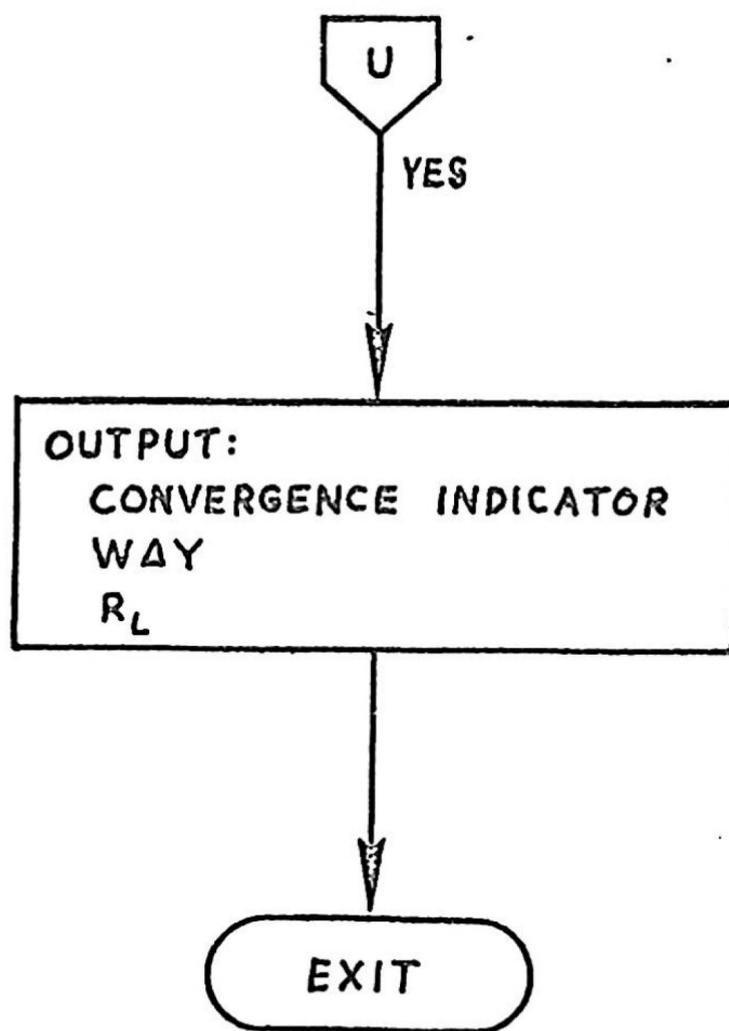




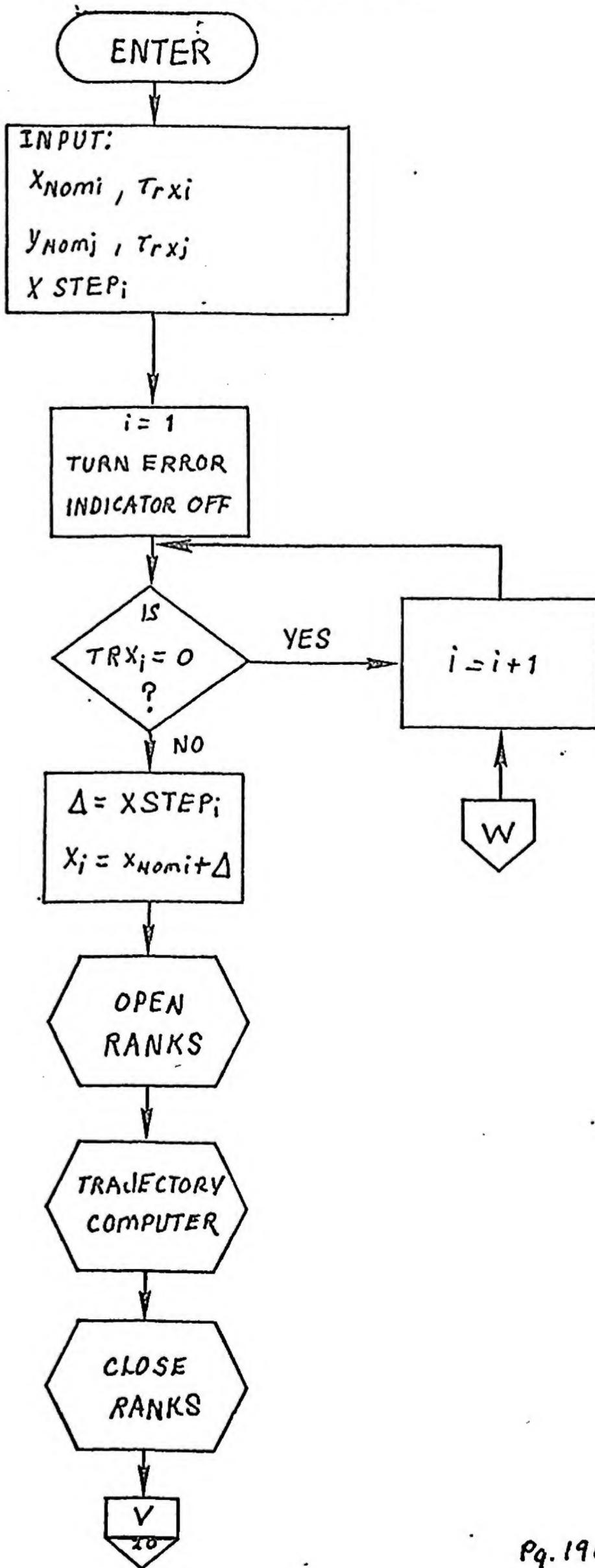
## RESIDUAL COMPUTATION AND CHECKING

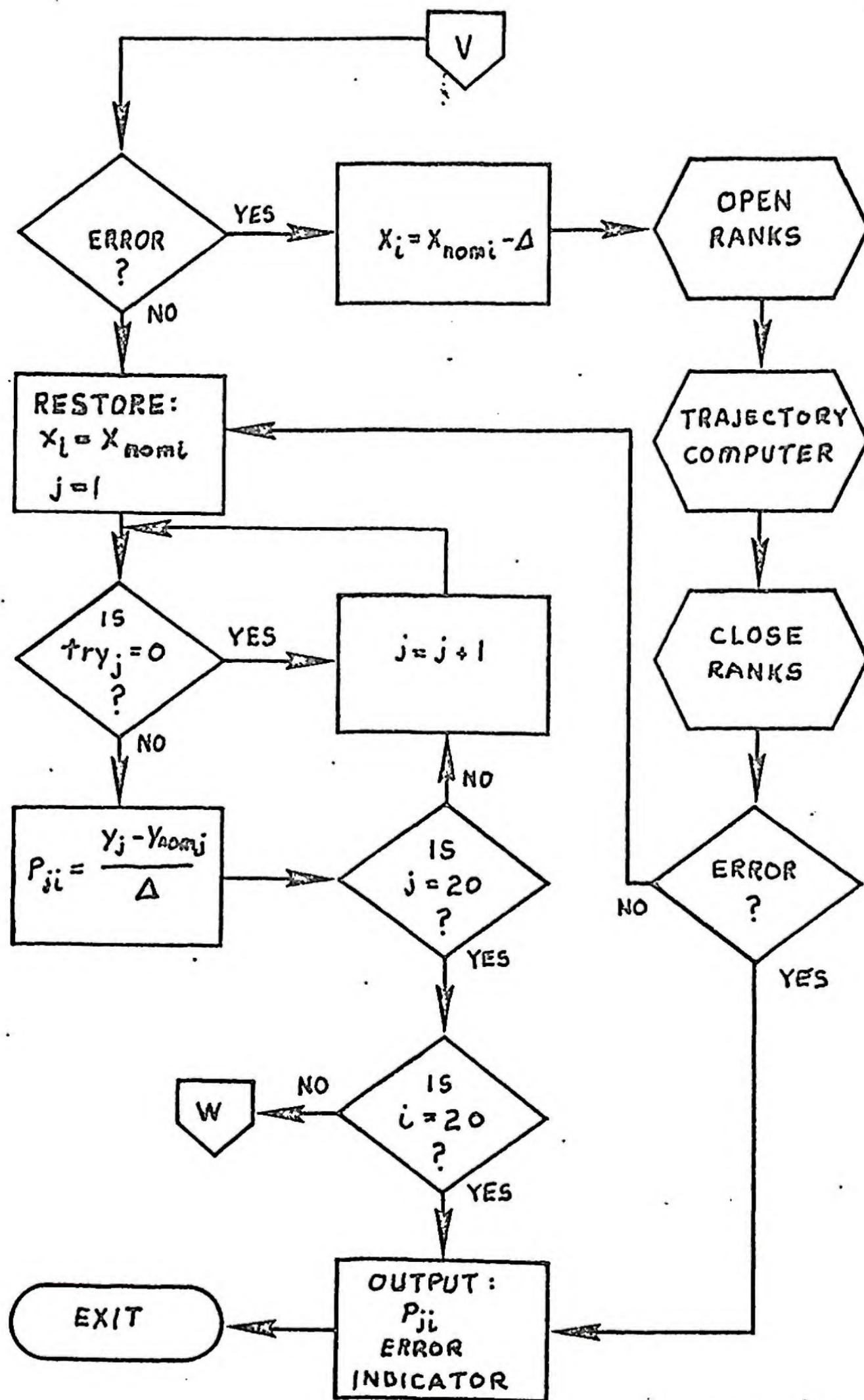






PARTIAL DERIVATIVE CALCULATION





## REFERENCE

:

1. Campbell, J. H.; Moore, W. E.; and Wolf, H.: A General Method for Selection and Optimization of Trajectories. Progress in Astronautics, Vol. 17, Academic Press, Inc., (New York), 1966.