THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DIGITAL SIMULATION OF THE HOMEOSTAT MODIFIED TO SHOW MEMORY AND LEARNING

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF PHILOSOPHY

by

BARNEY LEE CAPEHART

Norman, Oklahoma

1967

DIGITAL SIMULATION OF THE HOMEOSTAT MODIFIED

TO SHOW MEMORY AND LEARNING

APPROVED BY

DISSERTATION COMMITTEE

ACKNOWLEDGEMENTS

The author wishes to acknowledge his indebtedness to Dr. Richard A. Terry whose guidance, patience and understanding made possible the success of this investigation, and also to all members of the author's advisory committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER I

INTRODUCTION

The merits of investigating self-organizing systems are widely recognized. One of the earliest and most well-known theories of self-organizing systems is W. Ross Ashby's concept of ultrastability. In his book, Design for a Brain,[1] Ashby presents the concept of ultrastability as the mechanism used by the human nervous system to produce adaptive behavior. The use of this concept is illustrated by a machine, actually a special purpose analog computer, called the Homeostat. The Homeostat has been recognized and accepted by the bionics community as a valuable contribution to the field of modelling self-adaptive systems.

The Homeostat itself is a hardware device. Thus, any attempt to use a Homeostat in new self-adaptive systems research work would require considerable initial cost and effort to build the device. An even greater effort could result if substantial modifications of the device

---

[1] W. Ross Ashby, Design for a Brain (John Wiley and Sons, Inc., New York, 1960).

were made to incorporate new concepts of operation. One of the goals of this paper is to describe a procedure to simulate the Homeostat on a large scale general purpose digital computer. This would preclude the cost and effort of any hardware construction and subsequent modification. Portions of Ashby's work will be reproduced to show that the behavior of the computer model is identical to that of the actual Homeostat.

Ashby himself points out the major faults of the concept of ultrastability in its failure to adequately describe the behavior of the nervous system. These faults are 1) the exponential increase in time required to adapt for a system of increasing size; 2) failure to accumulate adaptations; that is, there is no memory function; and 3) failure to show reduced adaptation time to repeated inputs; that is, there is no learning function. Another goal of the paper is to modify the computer simulation of the Homeostat to incorporate the functions of memory and learning into its behavior. Examples of this new behavior will be shown.

A secondary goal of this paper is to promote enthusiasm for Ashby's concept of ultrastability. Since the publication of Design for a Brain in 1952, there has been some interest in the Homeostat as a device; but there has been little interest in the idea of ultrastability itself. It is the opinion of this author that the concept

of ultrastability has much unappreciated merit. Hopefully, the removal of two major faults will result in a greater acceptance of Ashby's pioneer work.

CHAPTER II

THE ULTRASTABLE SYSTEM

This chapter is a summary of the ideas leading to Ashby's concept of ultrastability. The material is taken wholly from the first seven chapters of Design for a Brain.[2]

According to Ashby, the free living organism and its environment, taken together, may be represented with sufficient accuracy by a set of variables that forms a state determined system. Since the organism affects the environment, and the environment affects the organism, we have a system with feedback. In block diagram form

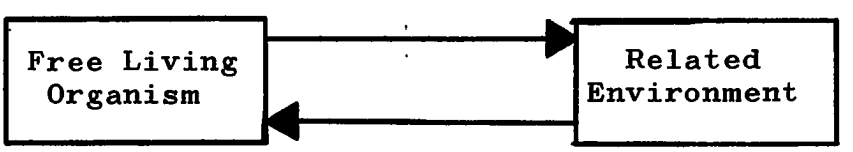


Figure 1 - System with feedback

Of the many variables associated with a living organism, a certain set can be selected and labeled as essential variables. Every species has a number of

---

[2] Ibid.

variables which are closely related to survival, and which are dynamically related so that significant changes in any one eventually leads to significant changes in the others. These important and closely related variables of the organism will be referred to as the essential variables.

Examples of essential variables in some animal might be:

(1) amount of oxygen in the blood

(2) pulse rate

(3) body temperature

We have previously mentioned that the system composed of an organism and its environment contains feedback. The presence of feedback immediately requires us to consider the stability of the system. There are many definitions and many different types of stability which could be discussed. However, since we are dealing with a state determined system, we can define stability in terms of a given trajectory in state (or phase) space. Given a phase space diagram of a state determined system, and a region of the phase space, the region is said to be stable if the trajectories from all points in the region stay within the region. If all trajectories are stable, then the system is said to be stable.

Every stable system has the property that if it is displaced from a state of equilibrium, the resulting

behavior is such that the system is returned to the state of equilibrium. A variety of inputs will therefore produce a variety of matched outputs. An important feature of a system's stability is that it is a property of the whole system, and cannot be assigned to any specific part of it. Thus, the presence of stability implies some coordination of the actions between the parts. However, for a whole dynamic system to be in equilibrium at a particular state, it is necessary and sufficient that each part should be in equilibrium at that state, in the conditions given to it by the other parts. Also, if the state of equilibrium is called a goal, then the stable system is "goal seeking" if displaced from the goal.

Based on the preceding principles, Ashby proposes the following definition of adaptive behavior--[3]

> a form of behavior is adaptive if it maintains the essential variables of the organism within physiological limits.

Adaptive behavior and stability can now be related by recognizing that--[4]

> adaptive behavior is equivalent to the behavior of a stable system, the region of stability being the region of phase space in which all of the essential variables lie within their normal limits.

Now, to be adapted, the organism, guided by information from the environment, must control its essential

---

[3] Ibid., p. 58.

[4] Ibid., p. 64.

variables, forcing them to stay within the proper limits, by so manipulating the environment (through its motor control of it) that the environment then acts on them appropriately. This implies a process of trial and error since no apriori information about the environment is assumed. The final behavior of the organism will depend on the outcome of the trials, i.e. how the essential variables have been affected. Thus, in addition to the feedback shown in Figure 1 previously, a second feedback path as shown in Figure 2 must exist.

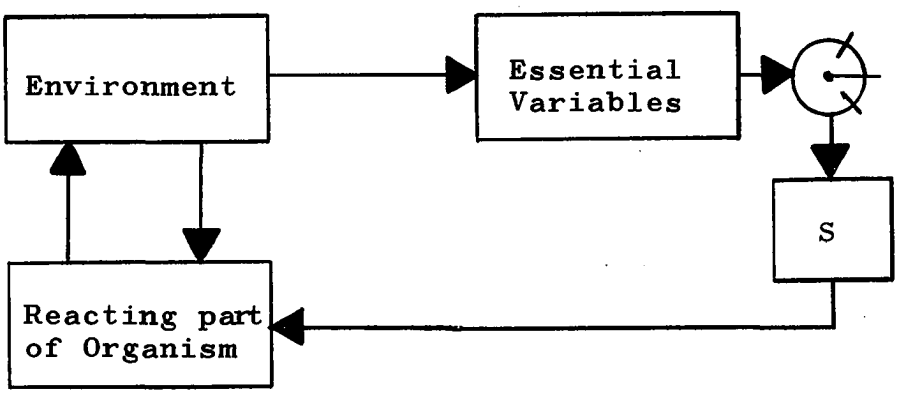


Figure 2 - System with Ultrastable Feedback

In Figure 2, the second feedback path is through the essential variables to a gating mechanism, S, which selects which general reaction shall occur. The first feedback loop consists of ordinary sensory input. The second feedback loop goes through the essential variables and carries information about whether the essential

variables are, or are not, outside their normal limits, and affects the gating mechanism S. In a nutshell, the first feedback plays its part within each reaction, the second feedback determines which reaction shall occur.

The basic rule for adaptation by trial and error is--if the trial is unsuccessful, change the way of behaving; when and only when the trial is successful retain the way of behaving. Regarding the system depicted in Figure 2, the basic rule is equivalent to the following formulation:

(1) When the essential variables are not all within their normal limits (i.e. when the trial has failed), no state of S is to be equilibrial.

(2) When the essential variables are <u>all</u> within normal limits then every state of S is to be equilibrial.

The gating mechanism S, is thus seen to be the dominant factor in the ability of the structure of Figure 2 to show adaptive behavior. S is assumed to produce step function changes, which in turn alters the basic behavior pattern of the reacting part of the organism. Since S is a mechanism showing a step function as its main characteristic, it may conveniently be referred to as a step-mechanism.

For any state determined system, the behavior of a variable at any time depends on the value of that particular variable, and all other variables at that time. Thus,

given a state determined system with a step mechanism at a particular value, all the states with the step mechanism at that value fall into two categories, those whose occurrence does not produce a change in the step-mechanism's value, and those whose occurrence produces a change in the step-mechanism's value. The latter states are the critical states of the system; and the step-mechanism will change its value should one of them occur. We assume that the only variables that show step function changes are those in S. The variables associated with the environment and the reacting part of the organism, including the essential variables, are assumed continuous.

We now have sufficient background to present the concept of an ultrastable system, as stated by Ashby.[5]

> Two systems of continuous variables (that we call "environment" and "reacting part of the organism") interact, so that a primary feedback (through complex sensory and motor channels) exists between them. Another feedback, working intermittently and at a much slower rate, goes from the environment to certain continuous variables which affect some step-mechanisms, the effect being that the step mechanisms change value when and only when these variables pass outside given limits. The step-mechanisms affect the reacting part of the organism, and by acting as parameters to it, they determine how it shall react to the environment.

Since the ultrastable system is stable by definition, we can use the concept of ultrastability to model the adaptive behavior of an organism.

Any given organism will have a finite set of

---

[5] Ibid., p. 98.

essential variables and critical states. If the organism is confronted with an environmental state to which it was not previously specifically adapted, then adaptation, if it can occur at all, must occur in the following way. The new situation acts to displace the variables of the reacting part of the organism from their state of equilibrium. The feedback from the environment to the reacting part of the organism is such that the resulting system is either stable or unstable with regard to the new environmental state. If the system is stable, the variables will all return to the state of equilibrium after some period of time; and the organism shows that its present state is adapted to the new environmental state. If the system is not stable, the essential variables will eventually exceed their limits, causing the system to reach its critical state. Reaching the critical state, the system acts to return the variables to acceptable values by making step mechanism changes. As long as any essential variable exceeds its limit, step mechanism changes will continue to occur. Only when a step mechanism reaches a value such that the system becomes stable again will the step mechanism changes stop. Thus, if adaptation is possible at all, the step mechanism will produce values eventually, which will make the system stable.

MATHEMATICAL TREATMENT OF ULTRASTABILITY

Let us consider a linear system described by a set of differential equations of the form $\dot{x} = Ax$ in vector-matrix form. For example, a second order servomechanism which is usually described by the equation

$$\ddot{x} + 2\zeta W_n \dot{x} + W_n^2 x = 0$$

can be written in the form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -W_n^2 & -2\zeta W_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

which is of the general form $\dot{x} = Ax$. A more general system of second order could be given as

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2$$

$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2$$

Taking a system described by these two equations, we could consider the requirements necessary to form an ultrastable system as proposed by Ashby. Since x_1 and x_2 are continuous variables of the system we could call them the essential variables. Limits must now be assigned to these variables which specify the critical states. These limits can be taken as x_1CRIT and x_2CRIT . To create the ultrastable feedback loop we must identify a certain parameter of the system to be altered if either x_1 or x_2 or both

ever reach or exceed their critical states. The coefficients of the describing equations can be taken as functions of a parameter called p. The equations become

$$\dot{x}_1 (x_1, x_2, p) = a_{11} (p) x_1 + a_{12} (p) x_2$$

$$\dot{x}_2 (x_1, x_2, p) = a_{21} (p) x_1 + a_{22} (p) x_2$$

If the value of p is altered as a result of one of the variables exceeding its critical state, then the coefficients $a_{1j}(p)$ will show step function changes in value.

Following Ashby, the above system can be discussed in terms of its behavior in the phase plane. Recall that a region of phase space is called stable if the lines, or trajectories, of behavior from all points within the region remain within the region. Now the effect of changing the parameter will be to alter the system field; and possibly alter the stability of the system.

As an example to specifically illustrate the behavior in the phase plane, consider a second order servomechanism with the value of damping taken as a function of the parameter p. We have

$$\dot{x}_1 = 0x_1 + 1x_2$$

$$\dot{x}_2 = -W_n^2 x_1 - 2W_n \zeta (p) x_2$$

Assume $x_1(0) = x_2(0) = 0.5$, $x_1\text{CRIT} = x_2\text{CRIT} = 1.0$; and that the damping is given as a function of p by the following table.

p	1	2	3	4	5
$\zeta(p)$	0	0.5	-0.5	2	-2

For each value of the parameter p, a different field or phase space picture, will result. A plot of the field for each value of p is shown below. In each case the system is started at the initial state $x_1(0) = x_2(0) = 0.5$.

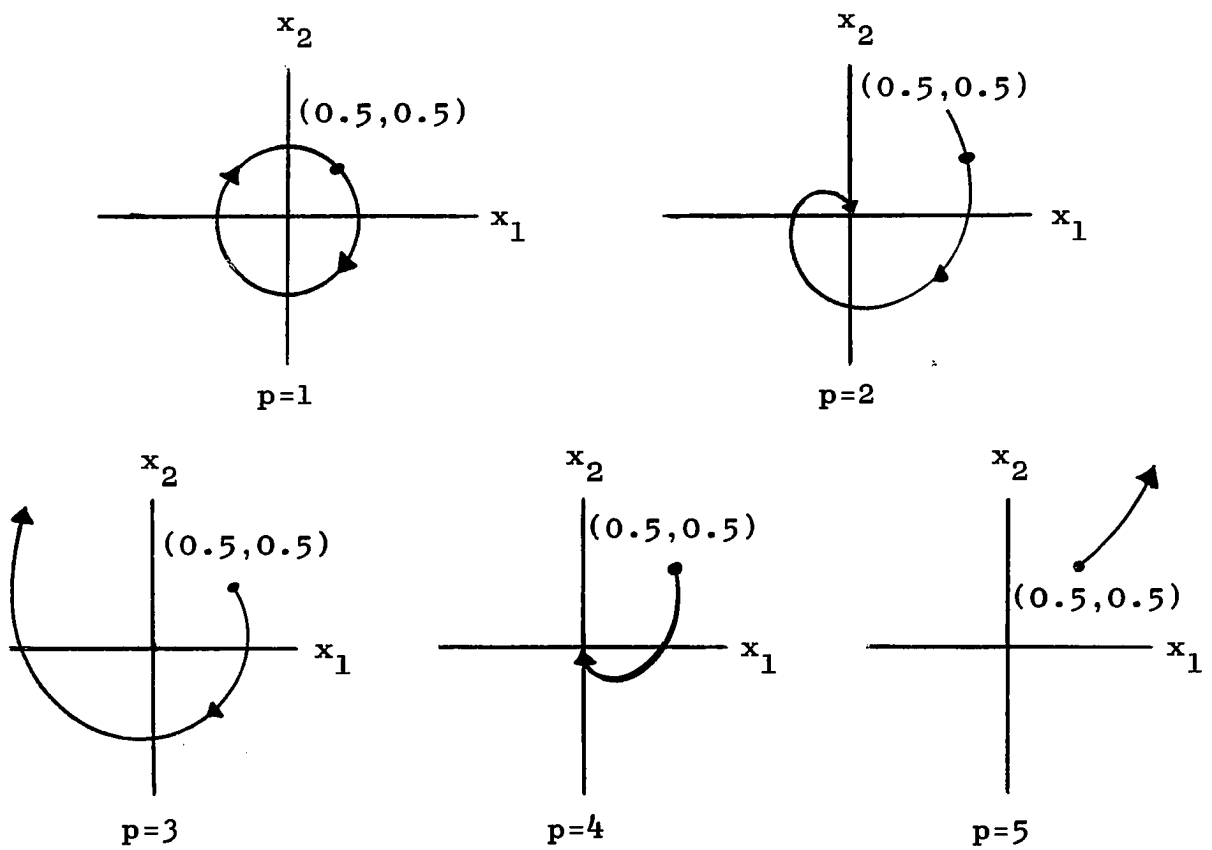


Figure 3

Phase plane behavior of a second order system with damping a function of p.

In the preceding graphs if $x_1 = x_2 = 0.5$ initially, then values of the parameter $p = 1, 2, 4$ will result in stable fields. Values of $0 = 3, 5$ will result in unstable fields. For the case $p = 3$, variable x_2 will exceed its critical state after some time; and for the case $p = 5$, variable x_1 will exceed its critical state.

To show the effect of ultrastability, consider a sequence of parameter values of $p = 3, 1, 5, 4, 2 \ldots$. If the system is started with $x_1 = x_2 = 0.5$ initially, and $p = 3$ initially, then the system will be unstable and variable x_2 will eventually exceed its critical state. As a result a step change will occur which changes p to 1. For this case, a steady state oscillation will exist with constant amplitude. Thus, x_2 will again exceed its critical state about one half cycle later. This will result in a step change causing p to become 5, and the system is unstable. Variable x_2 will again have exceeded its critical state causing a step change and producing $p = 4$. With this value of p the system is stable, and both x_1 and x_2 will decay exponentially to the equilibrium point $x_1 = x_2 = 0$. Adaptation has now occurred, and the value of p will remain equal to 4 until some new input occurs which causes the system to become unstable. This sequence of events showing ultrastability is pictured below.

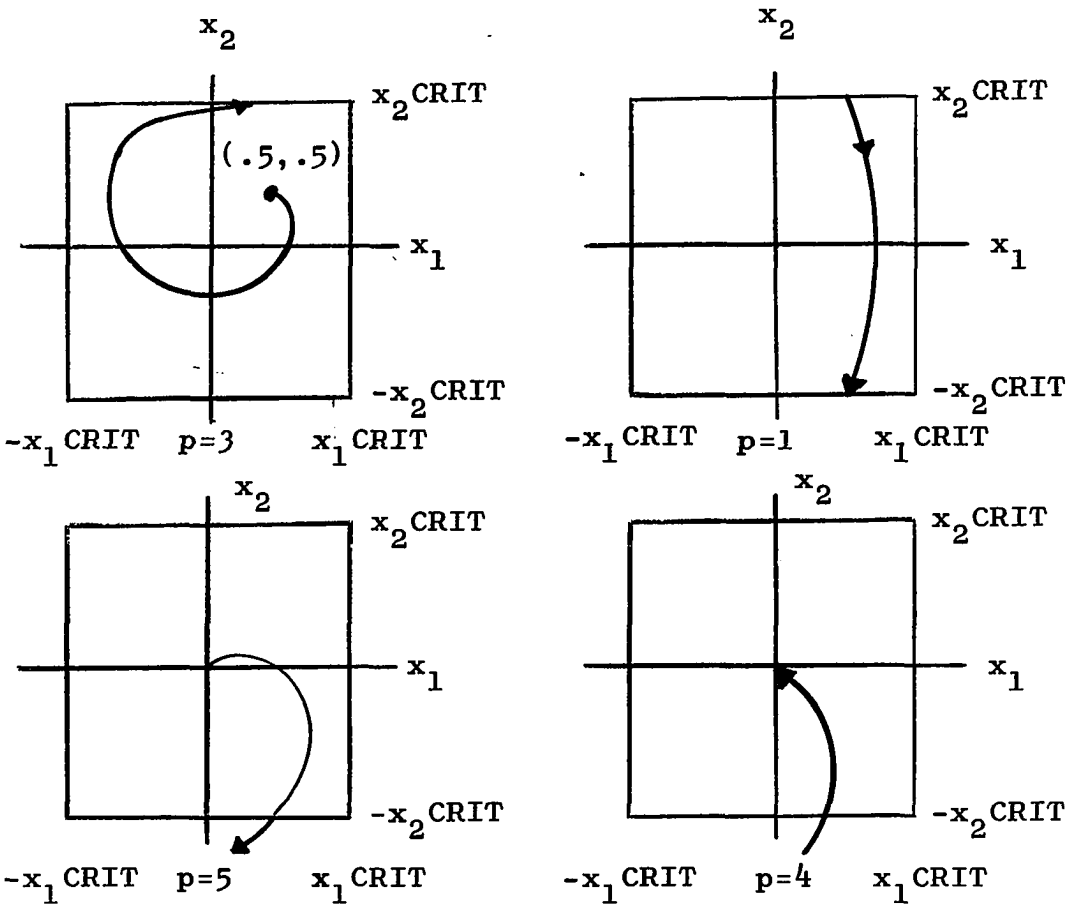


Figure 4

Phase plane behavior of an ultrastable system.

In the next chapter, the concept of ultrastability is applied to the design of an analog computer-type device, called the "Homeostat." The Homeostat exhibits a certain amount of adaptive behavior, but it has some very basic limitations. Examples of its abilities and limitations follow the details of its construction.

CHAPTER III

THE HOMEOSTAT

Description of the Homeostat

The Homeostat is actually a type of analog computer. It is composed of four units, each containing a D'Arsenval meter movement driving an output potentiometer. The angular positions of the meter movements represent the four continuous variables of the system. Each meter coil actually has four windings, to which the four variables are connected through variable sign/gain feedback elements. Thus, every output is connected to every input.

A block diagram of one of the four identical units of the Homeostat is shown below. For this unit, x_1 is the associated variable, and is given as the output position of the potentiometer. Variable x_1 is multiplied by the variable sign/gain feedback element b_{11}, and fed back to the input. The remaining variables, x_2, x_3, and x_4 are each fed into this unit through variable sign/gain feedback elements b_{12}, b_{13}, and b_{14} respectively.

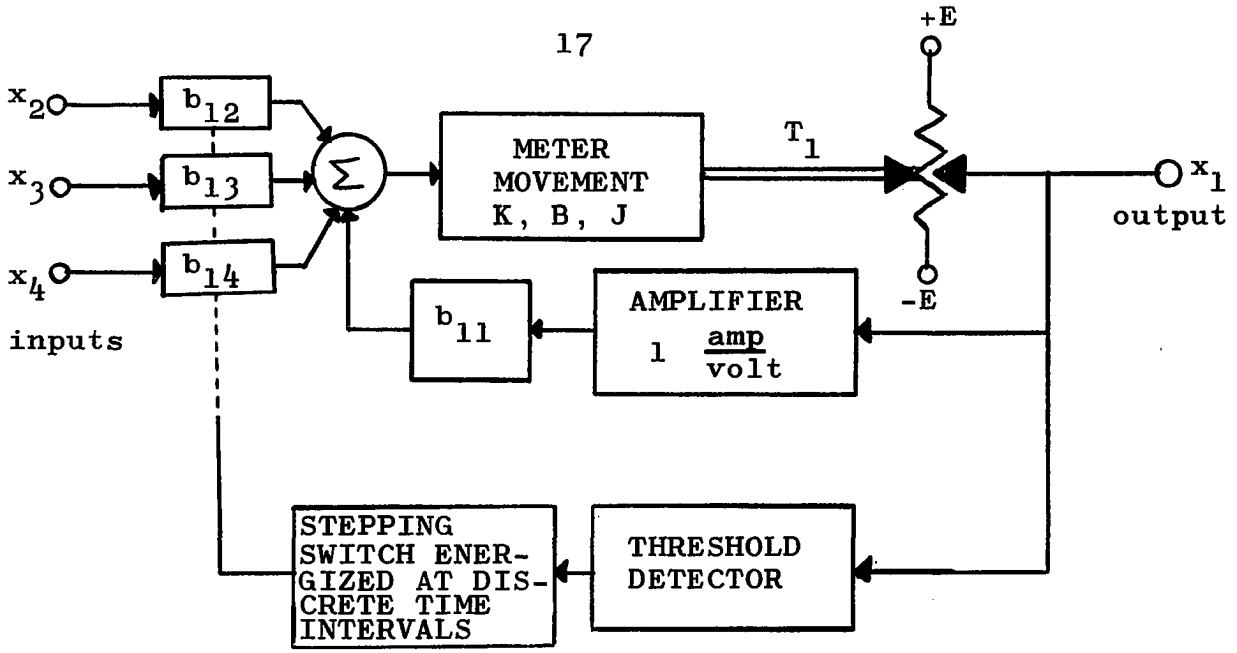


Figure 5

Block Diagram of one unit of the Homeostat.

At this point we have described only a dynamic system that could be stable or unstable, depending on the values of the sign/gain feedback elements. Recalling the concept of an ultrastable system, we see that our system, so far, has only the first feedback path; and does not have the "ultrastable" feedback path. We can introduce the second feedback path by calling the four variables essential variables, adding a device to sense whether they are within their normal limits; and if they are not, automatically changing the values of certain of the sign/gain feedback elements. In the Homeostat, Ashby has selected, for each unit, the three "outside variable" inputs to contain the automatically adjusted sign/gain feedback elements.

The "self feedback" of each unit is left for manual adjustment of sign/gain. A detailed discussion of the setting of the self feedback values is presented in a later chapter.

For any given unit, the variable associated with that unit is an essential variable. If that essential variable exceeds its normal limits, then the three sign/gain elements feeding back the other three variables should automatically show step function changes in their values. In each unit of the Homeostat, this process is accomplished by using a threshold detector, and a stepping switch to change the sign/gain elements.

The addition of the second feedback path has transformed our system into an ultrastable system; and now it can be used to demonstrate some of the principles of adaptive behavior.

Derivation of the Homeostat Equations

We can use the block diagram shown in Figure 3 to determine the set of differential equations describing the behavior of the Homeostat. Figure 5 shows one of the four units which are interconnected to form the Homeostat. Using D'Alembert's principle we can equate torques associated with the meter movement. The torque, T_i, generated by the meter movement is given as

$$T_i = K_i \left[b_{i1} x_1 + b_{i2} x_2 + b_{i3} x_3 + b_{i4} x_4 \right]$$

and this must equal the sum of the torques of the mechanical components of the meter movement:

$$T_i = J\ddot{x}_i + B\dot{x}_i + Kx_i$$

Thus,

$$J\ddot{x}_i + B\dot{x}_i + Kx_i = K_i \sum_{j=1}^{4} b_{ij}\, x_j$$

For the case where the moment of inertia, J, of the meter movement is negligible, the equation reduces to first order as:

$$B\dot{x}_i + Kx_i - K_i \sum_{j=1}^{4} b_{ij}\, x_j = 0$$

In state variable, or vector-matrix form, the equations are:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{B}(K_1 b_{11} - K) & \frac{K_1}{B} b_{12} & \frac{K_1}{B} b_{13} & \frac{K_1}{B} b_{14} \\ \frac{K_2}{B} b_{21} & \frac{1}{B}(K_2 b_{22} - K) & \frac{K_2}{B} b_{23} & \frac{K_2}{B} b_{24} \\ \frac{K_3}{B} b_{31} & \frac{K_3}{B} b_{32} & \frac{1}{B}(K_3 b_{33} - K) & \frac{K_3}{B} b_{34} \\ \frac{K_4}{B} b_{41} & \frac{K_4}{B} b_{42} & \frac{K_4}{B} b_{43} & \frac{1}{B}(K_4 b_{44} - K) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

By relabeling the constants, we can simplify the above set of equations to:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

or using general matrix motation:

$$\dot{x} = Ax$$

where the a_{ij}'s determine the sign and magnitude of each variable fed back. To solve this set of equations it is necessary to know the set of coefficients, a_{ij}, and the initial values of the variables $x_1(0)$, $x_2(0)$, $x_3(0)$ and $x_4(0)$. The a_{ij}, $i \neq j$, terms are taken as the random value components automatically selected by the movement of stepping switches. The a_{ij}, $i = j$, terms are taken as the self feedback elements, and are externally controlled.

The Homeostat equations, and the associated threshold detection and switching mechanisms, are amenable to solution using digital simulation. In the next chapter the simulation procedure is designed, and a FORTRAN program for simulating the Homeostat on the IBM 1410 is given.

CHAPTER IV

A FORTRAN PROGRAM FOR SIMULATING THE HOMEOSTAT

Since the Homeostat is described by a set of equations in the form $\dot{x} = Ax$, there is little difficulty in mathematically extending the size of the device. However, large quantities of specialized components would be required to actually construct a Homeostat with many variables. Here then, we see a problem where an application of some type of simulation technique could save the time and cost of building an actual device.

Simulation of the Homeostat by any given system would require basic mechanisms to perform the following tasks:

(1) solve a set of first order linear differential equations

(2) sense whether the value of an essential variable is greater than its normal limit (i.e. threshold detection), and

(3) switch-in new random value sign/gain elements for a certain unit upon detection of its essential variable exceeding the normal limit.

Since the problem requires the solution of a set of simultaneous differential equations, an analog computer would seem to be the logical choice. The size of the

Homeostat simulated would be limited only by the size of the actual analog computer used. However, a fairly complicated threshold detection and switching mechanism would have to be built and connected to the analog computer. Thus, the use of an analog computer would not completely relieve the necessity of large quantities of specialized components to simulate a Homeostat with a large number of variables.

To preclude the cost and effort of constructing any hardware at all, a large scale general purpose digital computer could be used. If the computer has a reasonable collection of library subprograms and functions, in addition to a FORTRAN or ALGOL compiler, the required programming task is somewhat simplified. A Runge-Kutta routine can be used to solve the set of differential equations; and a random number generation routine can be used to select the random sign/gain feedback elements. The threshold detection function can be accomplished easily with several FORTRAN or ALGOL statements. One of the advantages of the digital computer simulation is that in addition to eliminating the need for an initial outlay of hardware, changes in the computer model can be made without hardware modifications or additions. However, some seemingly minor changes in model behavior may require extensive programming effort. Thus, if a suitable digital computer and software system is available, an expanded and/or modified Homeostat

can be simulated and tested.

Since an IBM 1410 computer with a FORTRAN compiler was available, the digital simulation technique was chosen. The only modification made in the computer model was to expand the size of the Homeostat so that up to ten variables could be used. No other changes in the operation of the Homeostat were permanently made. However, a number of optional changes for experimentation purposes were built in. These features can be used by changing paramenters on certain of the input data cards. One of these features is the ability to restrict the main diagonal terms of the A matrix (representing the self feedback terms) to negative values. This greatly increases the chance of any A matrix being stable. The details of the reason for including this feature are discussed in the next chapter.

The following list represents the initial design requirements of the computer program to simulate the Homeostat.

1. Allow up to ten units (i.e. variables); and read a card, N, specifying the exact number of units for each run. $N_{max} = 10$.
2. Read an initial condition vector that specifies information analogous to initial meter deflections of the Homeostat.
3. Provide for internal generation of the NxN matrix of coefficients (the A matrix). Read a card, K,

which specifies whether the A matrix will be generated internally, or whether the A matrix will be read in initially. If the A matrix is generated internally, still allow the diagonal terms to be read in. Read a card, K1, to determine whether all or part of the A matrix coefficients are to be generated internally. If the diagonal terms are to be generated internally, read a card, K2, to determine whether the diagonal terms shall have random signs or all negative signs.

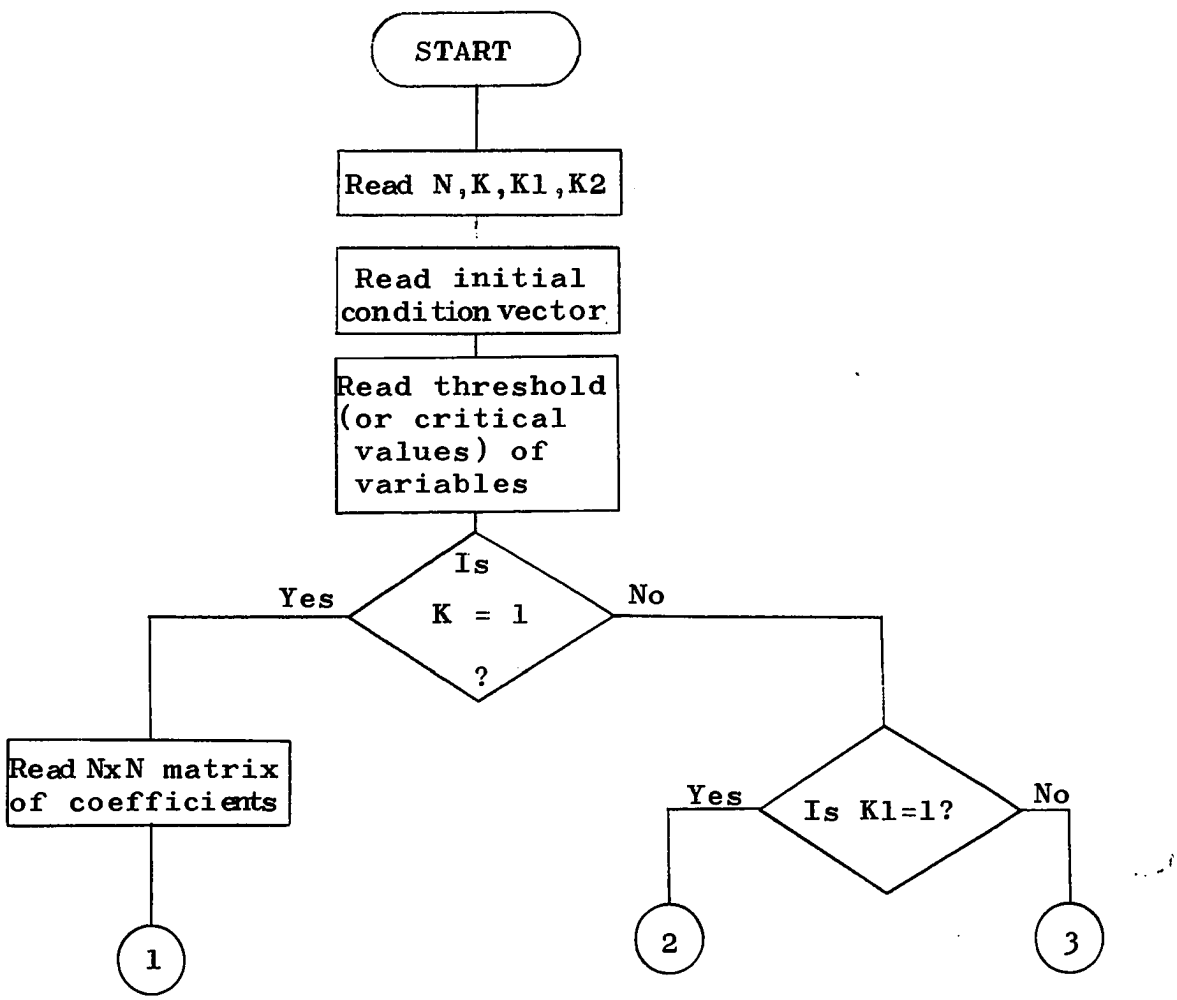
4. To follow Ashby's design, the terms of the A matrix should be integers from -9 to +9.
5. Once the A matrix is read in, or generated internally, test it for stability. Call a subroutine to find the characteristic equation of the A matrix. Call another subroutine to apply the Hurwitz test to the characteristic equation. If the A matrix is stable, do not solve the N differential equations. Record that stable state occurred.
6. If the A matrix is unstable, set up N simultaneous differential equations, and call an R-K-Gill subroutine to solve them.
7. Read threshold values of the N variables. If variable x_i exceeds its threshold, generate a new set of coefficients (of the A matrix) for a_{i1} to a_{iN}, except leave a_{ii} as it was; and continue the

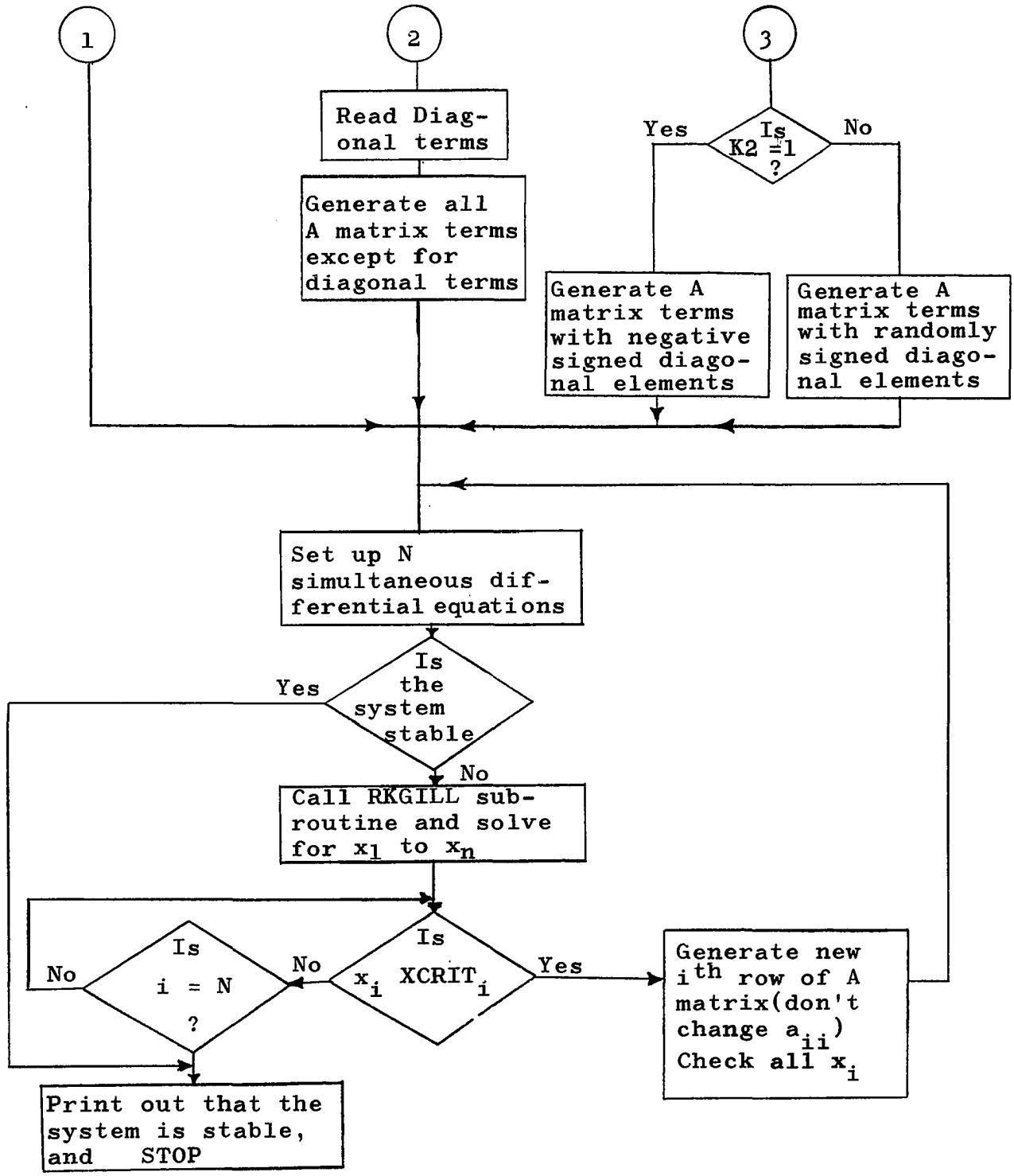
solution of the equations.

8. Record the number of step mechanism changes (i.e. new A matrices) that were required to reach a stable state.

9. Since a stable state might not occur at all, or not in a reasonable length of time, provide a means for the program to stop after a certain number, I1O, of step mechanism changes. Record that stability was not achieved. Using these design considerations, a flow chart of the computer program was constructed, and is shown on the next page.

FLOW CHART FOR COMPUTER PROGRAM
TO SIMULATE THE HOMEOSTAT




1
2
3
Read Diagonal terms
Generate all A matrix terms except for diagonal terms
Yes
Is K2 = 1 ?
No
Generate A matrix terms with negative signed diagonal elements
Generate A matrix terms with randomly signed diagonal elements
Set up N simultaneous differential equations
Is the system stable
Yes
No
Call RKGILL subroutine and solve for x_1 to x_n
Is x_i XCRIT$_i$
Yes
No
Is i = N ?
No
Generate new i^{th} row of A matrix(don't change a_{ii}) Check all x_i
Print out that the system is stable, and STOP

A FORTRAN program was coded in accordance with the above flow chart. The listing of this program is found in Appendix 1, along with a detailed flow chart.

The next chapter summarizes the work presented so far; and discusses some of the problems and limitations of the Homeostat.

CHAPTER V

EXAMPLES OF HOMEOSTAT BEHAVIOR

For a first example, consider the Homeostat with four variables; and let the purpose of this example be simply to show the ultrastable operation of the Homeostat. The four variables are given initial stationary values $x_1(0)$, $x_2(0)$, $x_3(0)$ and $x_4(0)$. A matrix of random digits is produced, and tested for stability. If the A matrix is stable, then no step changes are required, and adaptation has occurred. However, if the A matrix is not stable, the solution of the four equations is started. At some later time, due to the condition of instability, at least one of the variables will exceed its critical value. At that point, a step mechanism change will occur; and a new A matrix will result. If this A matrix is still unstable, then another step change will soon occur, and another new A matrix will result. This process will continue until a stable A matrix is obtained, and adaptation occurs; or until some prescribed time limit is exceeded, and failure to adapt is recorded.

This basic example was run on the computer

simulation of the Homeostat, with the results shown graphically below.

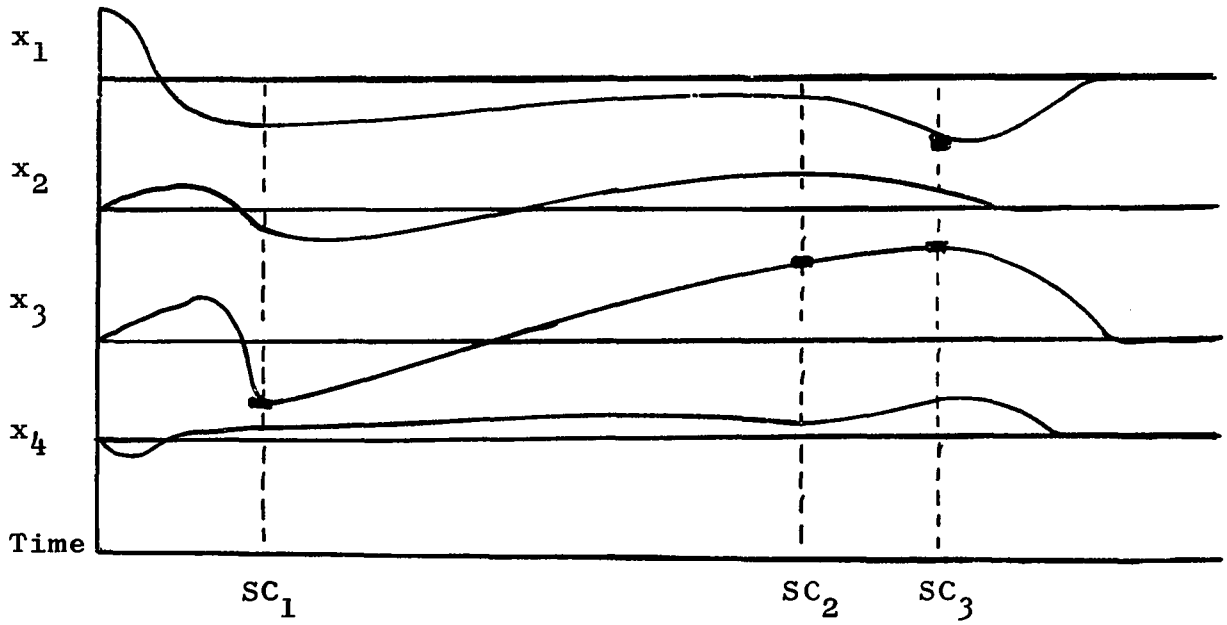


Figure 6

Example showing the ultrastable operation of the Homeostat.

Initially, the simulation was started with $x_1(0) = 0.5$ and $x_2(0) = x_3(0) = x_4(0) = 0$. At first, the system was unstable and x_3 exceeded its critical state, causing step change one. The resulting system was still unstable, and variable x_3 again exceeded its critical state, causing step change two. Again an unstable system resulted, and this time both x_1 and x_3 exceeded their critical states. This third step change resulted in a stable system finally, and the four variables returned to values of zero indicating that a stable state had been

reached. For any future trial with initial value of the variable given, the system will return to equilibrium, showing that adaptation to environment has occurred.

This example illustrated the basic ability of the Homeostat to change its internal behavior so that a stable state can be reached. This self-organizing behavior is a result of random variations in the system structure produced by the step mechanism changes.

As a second example, consider the Homeostat with only two variables. Let x_1 represent the reacting part of the organism, and x_2 represent the environment. Assume that some environmental state acts to displace x_1 and x_2 from equilibrium to new values, $x_1(0)$ and $x_2(0)$. In addition, the environment is assumed to show some property of alternation in its effect on the reacting part of the organism. Adaptation of the Homeostat to this environment could be viewed as analogous to the adaptation shown by a baby that exibits the same sucking reflex when presented either with a bottle or a pacifier. In this example, the alternation in the environment is produced by changing the sign of the self feedback element of the x_2 unit.

The Homeostat is started with x_1 and x_2 given the values $x_1(0)$ and $x_2(0)$. If the system is not stable initially, step mechanism changes will occur until stability is achieved. Next, the sign of the self feedback

element in the x_2 unit is reversed, and x_1 and x_2 are placed at $x_1(0)$ and $x_2(0)$ again. If the system is stable under this new change, then the system has adapted to the new environment. If the system stability is destroyed, step mechanism changes occur to restore stability. This process is repeated until the step mechanism changes produce a system which is stable under the reversal of the sign of x_2 self feedback. The computer simulation of the Homeostat was used to run this example, and the results are shown below in graphical form.

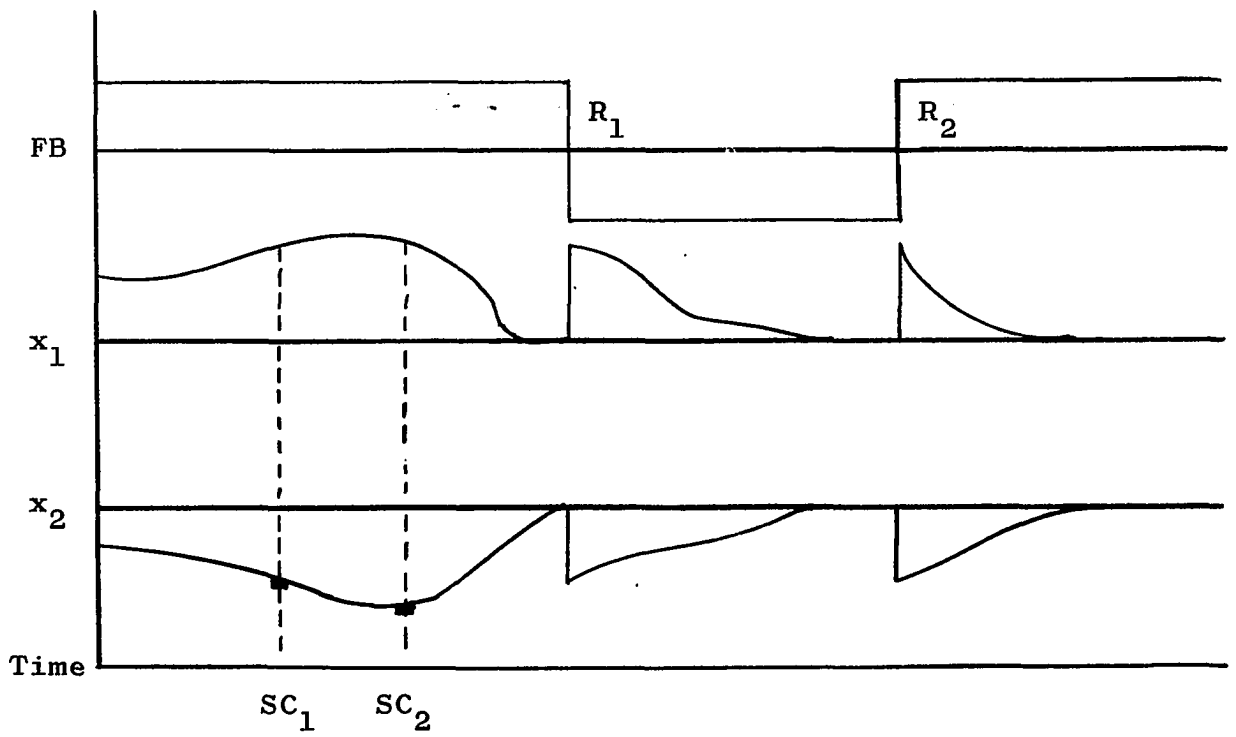


Figure 7

Record of computer simulations behavior when a feedback element FB was reversed from time to time.

This second example is similar to Ashby's example in Design for a Brain, pp. 115-116. At first, two step changes were necessary to even provide stability. However, the final state was then stable for reversal of FB at R_1 and R_2.

As a third example, consider the Homeostat with four variables; x_1 and x_2 representing the reacting part of the organism, and x_3 and x_4 representing the environment. Now suppose that the adaptation required is a form of training; e.g. we could require that a positive movement of x_1 be followed by a negative movement of x_2. In the event that x_2 moved positively, a trainer would force x_3 past its normal limit as a punishment. If the initial system is not adapted to this environment, then step changes will occur until a satisfactory system results. The computer simulation of this example produced the data shown graphically below.

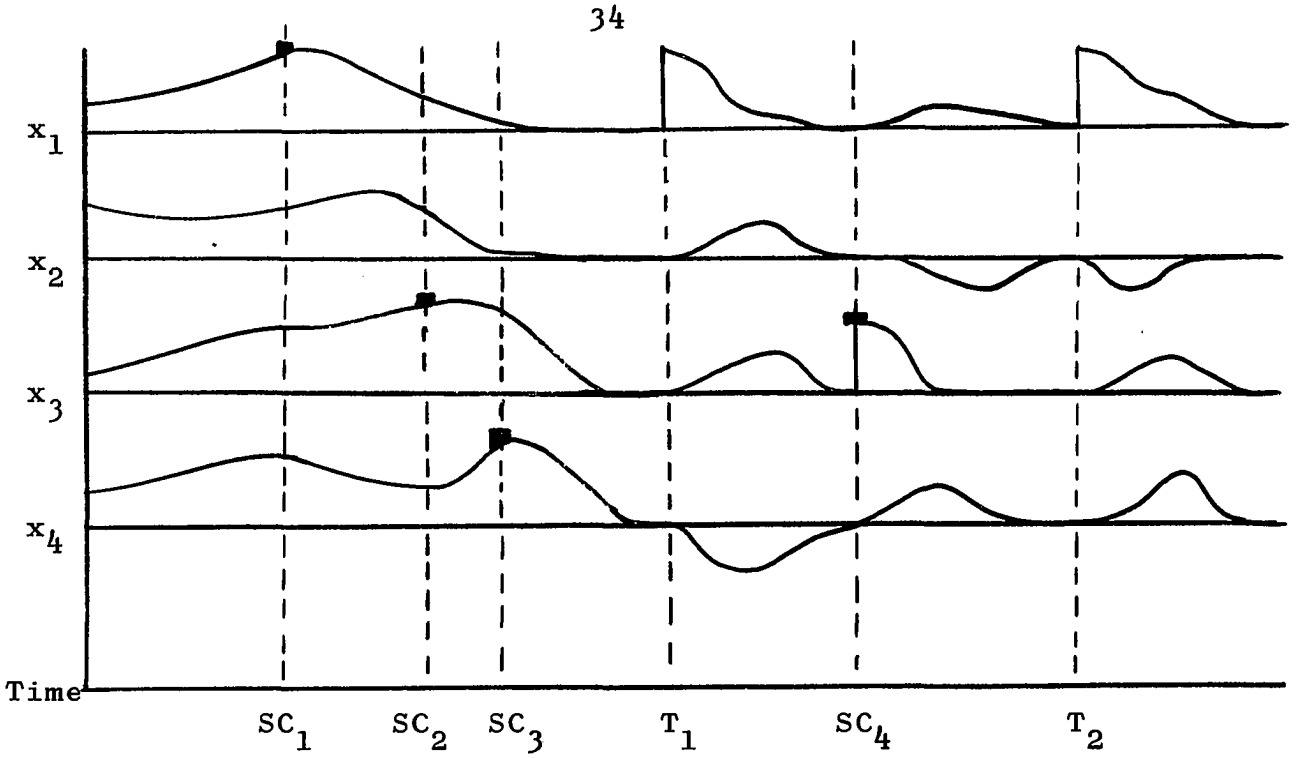


Figure 8

Adaptation to training. If x_2 moves positively, the trainer forces x_3 past its critical state as punishment.

This third example is similar to Ashby's example in Design for a Brain, pp. 113-115. The number of variables here was taken as four; whereas the original example had three. Note that initially three step changes were required to reach initial stability. At T_1, x_1 was forced positive, and x_2 moved positive, which was the undesired response. Variable x_3 was forced past its critical state as a punishment and a step change occurred. The resulting A matrix was stable, so at T_2, x_1 was again forced positive. This time x_2 responded by going negative, which was the desired response. The training was successful, and

for future trials, x_2 will always move negatively for a positive movement of x_1.

In the second and third examples, the results were identical in concept with those of Ashby. Since both the Homeostat and the computer simulation of the Homeostat use switching of random valued components exact duplicate results should not be expected. The same example run twice on either system would show a similar effect - equivalent end results, but different intermediate behavior of the variables. Next, results are given from a large number of runs of both examples on the computer simulated Homeostat.

So far we have presented the concept of ultrastability, the design of an ultrastable system called the Homeostat, and a computer program for simulating the operation of the Homeostat. In addition, several examples of adaptive behavior were run using the computer simulation of the Homeostat, and the results were shown. This work, then, shows primarily that the behavior of the Homeostat can be faithfully reproduced using digital computer simulation.

Ashby was limited in the actual examples of adaptive behavior of the Homeostat due to the restricted size of the device (four variables). However, the computer simulation expands the size of the Homeostat to ten variables and therefore several additional examples can be

given.

One of the major problems with the Homeostat is that the time required to adapt to a given situation increases exponentially with an increase in the number of variables. Ashby did not present any examples to verify or demonstrate this limitation. Using the computer simulation, examples of this time increase are easily shown. We can take two examples from the previous discussion, increase the number of variables, and run them again. This was done for the cases of number of variables N, equal to four and seven. The results are shown in tabular form in Table 1.

TABLE 1

A COMPARISON OF ADAPTATION TIMES AS THE NUMBER OF VARIABLES IS INCREASED

EXAMPLE	N	NUMBER OF RUNS	NUMBER THAT ADAPTED	AVERAGE TIME TO ADAPT	TOTAL TIME RUN
2	2	10	9	1.22 s. c.*	0.17 Hrs
	4	10	6	6.17 s. c.	1.40 Hrs
	7	5	2	20.50 s. c.	4.37 Hrs
3	4	9	6	2.83 s. c.	2.50 Hrs
	7	5	2	7.50 s. c.	7.09 Hrs
	10	9	0	- - -	17.15 Hrs

*s. c. = step changes.

From Table 1 it can be seen that the time to adapt (measured in terms of the number of step changes required to reach stability) is greatly increased for each case where variables are added. In some cases adaptation was not achieved at all within a fixed length of time. Also note the extreme running time as N becomes larger.

Ashby points out that the probability of stability associated with a given A matrix is greatly increased if the main diagonal terms are required to be negative.[6] The results presented in Table 1 were produced with all runs except for the case N = 2 made with all diagonal terms negative. To show the need for this restriction, examples two and three with N = 4 were run with randomly signed main diagonal terms. The results are shown in Table 2. Note that in most of the runs adaptation was not achieved at all in a fixed, reasonable length of time.

TABLE 2

RESULTS OF RUNS WHERE THE A MATRIX IS ALLOWED TO HAVE RANDOMLY SIGNED MAIN DIAGONAL TERMS

EXAMPLE	N	NUMBER OF RUNS	NUMBER THAT ADAPTED
2	4	9	1
3	4	9	0

[6] Ibid., pp. 258-260.

To justify the use of all negative main diagonal terms, we can turn to the large body of information on feedback control systems. The Homeostat can be considered as a collection of interrelated control systems, each representing a functional variable. The main diagonal terms of the A matrix represent the self feedback in each unit of the Homeostat. Comparing these units to well known position, velocity, and acceleration control systems, it is expected that the self feedback be negative, for negative feedback is the basic principle of operation of an error controlled system.

The above discussion points out one of the major problems in modelling adaptive behavior with an ultrastable system - the time to adapt becomes prohibitive as the number of variables in the system become large. The behavior of the Homeostat exposes two additional major problems - failure to accumulate adaptations (i.e. no memory), and failure to show reduced adaptation time for repeated inputs (i.e. no learning). These problems present some very severe restrictions to general acceptance of Ashby's work. It is the opinion of this author that the concept of ultrastability has much unrecognized merit; and that much can be accomplished to relieve some of the restrictions mentioned. In the next chapter, two problems are singled out for further investigation. They are the addition of memory; and the incorporation of

learning into the behavior of the Homeostat.

CHAPTER VI

DEVELOPMENT OF MEMORY AND LEARNING FOR THE HOMEOSTAT

Adaptive Behavior of the Homeostat

The Homeostat is a linear system whose behavior at any instant of time is specified by a set of first order differential equations in the form $\dot{x} = Ax$. Since the system is linear, the stability is determined wholly by the characteristic roots of the A matrix, and is independent of initial conditions given by the x(0) vector. For example, if the system is stable for an input of $x_1(0) = 0.5$, $x_2(0) = x_3(0) = x_4(0) = 0$; then it will be stable for an input of $x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0.7$. This assumes that an initial value of a variable is never taken greater than or equal to its critical value. If a variable was initially given a value exceeding its critical level, then a step change would occur; and possibly the stable system could be changed to an unstable one. With this restriction then, stability is independent of the initial values of the variables.

To what then, does the Homeostat adapt? Recalling

that Ashby defines adaptation as the process of seeking equilibrium or stability, we see from the above discussion that if the A matrix is stable for any one set of initial conditions, then it will be stable for all sets of initial conditions. Thus adaptation is generally not with respect to arbitrary initial conditions, but to relations between the variables of the system, or the structure of the A matrix. For example, the Homeostat can adapt to the constraint that a forced change in x_1 be followed by an opposite change in x_2. This relation can be achieved by forcing step mechanism changes to alter the A matrix until a suitable system configuration is obtained. One way to cause these step mechanism changes is to force a variable, or variables, other than x_1 and x_2 past the critical state each time an "incorrect" response occurs. This case was illustrated as example 3 in Chapter IV.

In the initial versions of the computer model of the Homeostat, separate programs were used for each of the different conditions required for adaptation. That is, a separate program was required for each of the three examples of Homeostat behavior given in Chapter IV. If the Homeostat is to serve as a model for a general adaptive process then some changes must be made in the computer program in order for it to respond to the varying types of input that will occur.

To generalize the operation of the computer model,

the input dependent adaptive logic from each of the three simpler programs was combined into one large program. However, now that the general adaptive logic was included in one program, a means was needed to determine which portion of that logic should be used for a given input. Thus, a need arises for some form of input pattern classification scheme, where each input pattern produces a unique pattern classification signal thereby calling the required portion of adaptive logic. The design of the input pattern classifier should be as simple as possible, in order to minimize the necessary program changes. However, the form of the inputs to the Homeostat are such that a very careful examination is required to guarantee that the significant aspects of the inputs are considered.

Input patterns to the Homeostat are not immediately recognizable as "normal" patterns, since the input does not appear all at once, or even in a simple time sequence. An example of an input pattern is taken from example 3 in the previous chapter (Figure 8).

A positive displacement of variable x_1 must be followed by a negative displacement of variable x_2, or else variable x_3 will be forced past its critical state as a punishment.

Note what is required to determine the full nature of the input: if the x_2 solution is observed, it must not follow the direction of the x_1 input. This requires

some time for solution, and a stable A matrix must result. If x_2 does follow x_1 then x_3 is forced past its critical state, and the process is repeated.

Thus it can be recognized that some coding scheme is necessary to tell the Homeostat at the start of an experiment the exact nature of the input. Also note that in the above example, we could consider the solution to be obtained by repeated application of a simpler pattern.

The problem then, is to take a class of inputs and find some method of expressing the input as a group of simpler inputs, if possible; and somehow encode the result for input to the Homeostat.

This coded input must be such that input patterns can be classed as members of say one of N possible categories. The Homeostat must then be capable of classifying inputs as they occur, and storing the pattern classification along with the resulting A matrix.

In order to code the input pattern in some manner, certain features of the pattern must be selected so that knowledge of these features is sufficient to uniquely describe the pattern. This process is called feature recognition. The following features were selected as being descriptive of a wide class of possible input patterns to the Homeostat.

(1) Which variables of the system are related?

(1a) How are they related?

(1b) What are the consequences of an incorrect relation?

(2) Is the structure of the A matrix altered?

(2a) What terms of the A matrix are altered?

(2b) How are they altered?

(2c) What are the consequences of incorrect behavior?

Although all of these features are utilized by the computer program in some manner, they are not all required in identifying the input pattern class. Based on the above input features, we can construct a simple four category pattern classifier using only features (1) and (2) from the above list. A much more comprehensive pattern classifier could be built, but four categories were chosen as a matter of program simplicity. A future addition to the program could be made to widen the spectrum of input pattern classes. The four pattern classes are specified as below.

Class	Features
1	No variables related, A matrix unaltered
2	Variables related, A matrix unaltered
3	No variables related, A matrix altered
4	Variables related, A matrix altered

The original or "basic" Homeostat program was modified to include the input pattern classifier and the combined adaptive logic. This version of the program was

then tested using the three examples from Chapter IV. All three examples ran successfully, showing proper operation of the integrated program. No results are shown here since the results from the previous examples were simply reproduced. A listing of this version of the program is found in appendix B.

Incorporation of the Memory Function

Once the elementary pattern recognition scheme has been added to the computer model of the Homeostat, a fairly simple memory function can also be added. The following list specifies the design requirements for this memory function.

(1) Modify the program to store the A matrix that produces adaptation to a given input. The A matrix should be stored in a section of memory reserved for adaptations to inputs in the pattern class of the original input. If memory space is available, it might be desirable to store all stable A matrices resulting from a particular run.

(2) Modify the program to start by reading A matrices from the pertinent section of memory, rather than initially generating them internally. If a given A matrix from memory is not successful in producing adaptation to the present input, then another A matrix is read and tried. An unsuccessful A matrix obtained from memory is not

altered and tried again; instead, it is simply returned to memory, and another completely new A matrix is called from memory.

(3) Prepare a "preprocessor" which will store a specified number of stable A matrices in memory prior to the first run of the main program.

Operation of the computer model of the Homeostat with memory would proceed in the following manner - initially no memory would exist in the system. Thus adaptation to the first input would of necessity be the result of trial and error behavior of the ultrastable feedback mechanism. Once adaptation to this input is obtained, the resulting A matrix is stored in a section of memory reserved for members of the respective pattern class. Further A matrices resulting from other inputs of this class will be stored sequentially. Next, consider the application of a second input. This input is first classified; and then a check is made to see if the memory section corresponding to that pattern class is non-empty. If that memory section is empty, then adaptation can only occur through use of ultrastability. If the second input happened to belong to the same class as the first input, then a non-empty memory section will be found. The first A matrix in this memory section (in this case, the only A matrix) is called and tested to see if it provides

adaptation to the new input. If the first A matrix is successful, then a record is made of the fact that adaptation occurred from a memory search. Should the first A matrix from memory be unsuccessful, the next A matrix in sequence in the memory section will be tested. This process is continued until either a successful A matrix is found from memory; or until the memory is exhausted in which case the ultrastable feedback mechanism is activated.

An important aspect of the operation of the memory function is that when an A matrix is called from a section of memory and tested, if the A matrix is not successful, it is simply returned to memory; no attempt is made to modify it in any manner. As a later sophistication of the program, a change could be made to assess the utility of a particular A matrix from memory and determine some procedure to alter the A matrix such that it will produce adaption to the given input. The problem here would be how to determine the manner in which to modify a given A matrix for a given input. The program changes required would probably be quite complex.

After a period of operation where many different inputs have occurred, a fairly large memory could be expected. Thus, as time of operation increases, the probability of adaptation from memory to any given input should increase. The actual value of this probability will

depend on the diversity of the different inputs. A large class of fairly similar inputs could be expected to produce a high probability of adaptation from memory; whereas a very diverse class of inputs might produce a low or near zero probability.

In the computer program, a finite memory size must be selected. For reasons of simplicity, each section of memory corresponding to a given input class was given 1/ number of classes of the total memory space. Thus, for the case here with four pattern classes, the memory was divided into four equal parts. Arbitrarily, a figure of 400 matrix storage locations was selected for the total memory. This gave 100 A matrix storage locations per pattern class. At this stage of development of the program, the finite, fixed memory space results in a loss of any A matrix desired to be stored once that particular section of memory is full. Later, a program feature will be discussed which allows a selective process to decide which A matrices to keep, and which to dump.

The behavior of the human memory function provides the basis for this design of the memory for the Homeostat. If a situation occurs where there is no memory of a relevant adaptation, then a process of trial and error is used by the person. If adaptation can be achieved in this manner, then some result related to this process is stored in memory. This stored result is associated with some pattern

related to the particular situation. In some future situation similar to the original situation here, the stored result will be recalled and applied. In this manner, it is conceivable how a young child might start with virtually no memory, yet end up with a substantial memory of adaptations after a period of only trial and error behavior.

The program modification required to enable successful A matrices to be stored in memory, and to provide for initial memory search are not extremely complex. A utility subroutine is used to store successful A matrices on the disk; another utility subroutine is used to read A matrices from the disk. Most of the program changes are in the form of indexing procedures to allow reading and writing A matrices into and out of the correct sections of memory for a given pattern class. The inclusion of the preprocessor to store a collection of stable A matrices before starting the system is mostly for test purposes. However, if many of these initial A matrices prove to be successful, then the adaptation time for a number of input patterns will be reduced. A possible human analogy to this procedure could be inherited or inborn ability to adapt to certain classes of input. The operation of the program will be tested with and without an initial memory of A matrices; and the overall time to adapt to a fixed number of inputs will be recorded.

An attempt will be made to estimate the effect of

the memory function by computing the probability of adaptation from memory for a certain set of inputs applied after a fixed training sequence. This training sequence will consist simply of allowing the memory of the computer model of the Homeostat to record adaptations to a fixed number of typical input patterns.

A flow chart and listing for this version of the program is not specifically provided due to the minor differences in this version and the final version given in appendix D. To demonstrate the effect of adding a memory function to the Homeostat, the following three computer runs were made. In the first run no initial memory of A matrices was provided. Out of 20 inputs there were 13 adaptations from memory.

TABLE 3

RESULTS OF COMPUTER RUN WITH MEMORY INITIALLY EMPTY

PATTERN CLASS 3	
Number of inputs	20
Number of adaptations by trial and error	3
Number of adaptations from memory	13
Number of failures to adapt	4

For the second run, an initial memory of 15 stable A matrices was provided. The number 15 simply represents

an arbitrarily selected value. The data from the first run was then used for this run. Then, out of 20 inputs there were 19 adaptations from memory.

TABLE 4

RESULTS OF COMPUTER RUN WITH MEMORY INITIALLY STOCKED

PATTERN CLASS 3	
Number of inputs	20
Number of adaptations by trial and error	1
Number of adaptations from memory	18
Number of failures to adapt	1

The third run was made using the data from run number one to produce an initial memory of 3 A matrices. Next, 20 inputs were applied after the memory had been stocked. Out of the 20 new inputs, 18 adaptations occurred from memory. Thus, following a fixed training sequence of inputs, we have 90 per cent of the new inputs adapting from memory.

TABLE 5

RESULTS OF COMPUTER RUN FOLLOWING INITIAL TRAINING SEQUENCE

PATTERN CLASS 3	
Number of inputs following formation of memory	20
Number of adaptations from memory	18
Probability of an adaptation from memory	.90

The above results show that addition of the memory function has reduced the adaptation time for a significant number of inputs by providing adaptation from memory search rather than by trial and error. Table 5 shows that after a fixed training sequence, we can estimate the utilization of the memory at 90 per cent.

Incorporating Learning into the Homeostat

The discussion of addition of the memory function presented in the previous section suggests a way to incorporate learning into the behavior of the Homeostat. Here, learning is taken as a decrease in the amount of time required for the Homeostat to adapt to a given input pattern which is repeated. Since the A matrix resulting from an input pattern of some kind represents a successful adaptation, that matrix should be stored in memory, and recalled as one of the first A matrices tried when a new input pattern of the same class occurs. Taking elementary reinforcement as a model for learning we can make a simple extension of the memory function modification to enable the Homeostat to show a form of learning. A "success tag" can be added to each matrix that is successful, and therefore stored in memory. Whenever the ultrastable system produces an adaptation by trial and error, the corresponding A matrix is stored in the section of memory reserved for the respective pattern class, with a value of 1 as its

success tag.

For a new input pattern the operation of the Homeostat with learning proceeds in a manner similar to that of the Homeostat with only memory; except that the A matrices with the greatest value success tags would be tried first. Each time an A matrix from memory is used to produce adaptation to a new input, that A matrix will have its success tag increased by one, up to a certain maximum value. When the value of an A matrice's success tag reaches this maximum, that A matrix is assumed to become a part of the permanent memory, and its success tag is no longer increased with use.

In the event that no A matrix stored in memory produced adaptation to a given input the basic ultrastable mechanism of the Homeostat would generate a new A matrix, if possible. This new A matrix would then be added to memory with a success tag value of one.

The combination of ultrastable behavior and memory with the above described search procedure should result in a total system that shows a reasonable form of learning. After a period of time any pattern class section of memory will have a collection of A matrices with success tags of varying values. If now, the initial memory search, given the next input pattern, is organized such that the A matrices with the largest valued success tags are tried first, then the probability is greatest that one of the

first A matrices will be successful. In general, adaptation time can be considered as being inversely proportional to the frequency of a particular input pattern. High frequency input patterns should have very low adaptation times, possibly equal to some fixed minimum time. Low frequency inputs could have fairly long adaptation times due to the lengthy search of all the contents of memory.

The problem of selective memory operation due to a finite size memory can be handled by using the success tags associated with stored A matrices. As the memory contents near the maximum, a search can be conducted for A matrices with low valued success tags. These A matrices could be deleted from memory without significantly reducing the effectiveness of the memory and learning functions. New A matrices could then be added to the memory until the next deletion process occurred. Any of the new A matrices that were not used again one or more times before the deletion process would be eliminated. Thus, for a fixed memory size, over a long period of time, the memory would contain only A matrices with the greatest valued success tags.

The program changes required to enable the modified Homeostat to show learning are very minor. An indexing system to keep a record of the number of times each A matrix is used must be added, and a sorting routine to arrange the A matrices sequentially in memory according

to their success tags must be provided. A detailed flow chart and listing for this version of the computer program is given in appendix D. This represents the final computer program, simulating the Homeostat with pattern recognition, memory, and learning. To demonstrate the learning ability of the modified Homeostat, two computer runs were made. The first run used the simulation of the Homeostat with only memory, while the second run used the simulation of the Homeostat with memory and learning. For the first run, records were made of the number of adaptations that occurred from memory. In addition, the number of A matrices read from memory before adaptation occurred were recorded. For the second run, the same number of adaptations from memory will occur, but the number of A matrices read from memory should be reduced. This decrease in memory search time can be attributed to learning being shown by the modified Homeostat.

TABLE 6

RESULTS OF COMPUTER RUN SHOWING OCCURRENCE OF LEARNING IN THE MODIFIED HOMEOSTAT

Data Input Number	Number of Memory Searches without Learning	Number of Memory Searches with Learning
5	2	2
6	1	2
8	3	3
9	2	1
11	1	2
12	2	1
14	2	1
15	2	1
16	3	3
17	2	1
18	1	2
21	2	1
22	3	3
23	2	1
25	1	2
26	2	1
27	2	1
28	2	1
29	2	1
30	2	1
32	2	1
33	2	1
34	2	1
36	2	1
37	3	3
38	1	2
39	2	1
40	2	1
41	5	5
42	2	1
43	2	1
44	2	1
49	5	5
50	2	1
34	73	56

TABLE 7

SUMMARY OF RESULTS IN TABLE 6

Average Number of Memory Searches Without Learning for 34 Inputs	2.15
Average Number of Memory Searches With Learning for 34 Inputs	1.65

The above results show that the incorporation of the learning technique has resulted in a faster adaptation to a number of inputs. An overall reduction in memory search effort is given in Table 7. In all honesty, it must be pointed out here that the results from other computer runs showed only marginal reduction in adaptation time or failed to show any decrease at all. The reason for this appears to be the small number of A matrices stored in memory, resulting in an inability of the search technique to produce any clear cut improvement.

Relation of Homeostat Operation to the Human Brain

Basic Homeostat

Operating with the concept of ultrastability, the Homeostat provides a basic model for the adaptive behavior of the human brain. The Homeostat assumes no apriori information; it adapts by pure trial and error. The only required mechanisms are those which allow sensing of an

unstable state, and randomly altering behavior patterns. The Homeostat is quite similar to a small child in its behavior. Initially, a child has no learned behavior; each adaptation is by trial and error. Adaptation is acquired quickly for simple situations, and slowly to very slowly for complex situations. However, the child soon surpasses the basic Homeostat by showing memory and learning. For example, the young child soon learns the routine of the mother, and conditioned responses occur. The simple Homeostat is incapable of learning even the simplest task, if learning is taken as a reduction in adaptation time when a certain input is repeated some number of times.

However, the basic Homeostat behaves similarly to a fully developed adult that is faced with an environmental input that is unrelated to the current forms of learned response patterns stored in the adult's memory. If the new input were related to or closely associated with some previously learned behavior, then the adult would show learning fairly quickly, due mainly to the process of association. Since the basic Homeostat cannot show significant learning of any form, it would provide a poor model of an adult brain using learning by association.

Homeostat with memory and learning ability

The basic Homeostat can be modified to show a form

of memory and learning. The learning is associated with a pattern classification scheme for a fixed class of inputs. Four input pattern categories were chosen initially. The first step in the operation of the modified Homeostat is to scan the input environmental state and classify it according to a simple decision procedure. The Homeostat proceeds to adapt to the environmental input by the process of trial and error. If adaptation occurs within a fixed length of time, the resulting A matrix is stored, along with tags indicating the pattern classification and number of times the given A matrix has been successful. This process is repeated for a large number of input patterns within a certain class. As soon as a reasonable memory is stored, operation of the modified Homeostat is switched so that now each input is classified and the memory is searched for A matrices which have been successful for inputs of the same class. Within the pattern class, the A matrix, or A matrices, with the highest number in the tag position indicating number of times successful, is called and tried first. If that A matrix does not provide adaptation, the A matrix with the next highest tag is tried and so on, until a successful A matrix is found in memory; or until the memory is completely searched. If the memory has been completely searched in that pattern class, then operation reverts to that of the basic Homeostat, and adaptation will be found by trial and error.

However, if a successful A matrix is found in memory that A matrix will have its success tag increased by 1, up to a certain maximum value.

With repeated inputs, this form of operation should provide a reduction in adaptation time. Note the similarity in this form of operation of the modified Homeostat and the logical operation of a human being in a typical problem environment. Given a certain problem, a logical first step would be to classify the problem. Whether the exact problem had been worked before is of little significance. Once the problem has been classified, the memory is searched for the technique to solve the problem. The technique tried first will probably be the one that has worked most often before on similar problems. If that technique is not successful, the next most successful technique is used, and so on, until a successful technique is found or until the memory is exhausted. If a technique in memory worked, a mental note is made of the fact that that particular technique was again successful. Physchologically, the memory of that technique is reinforced. After enough problems, the most useful techniques fail to be reinforced significantly. A saturation level has been reached; and the memory of those techniques can be considered permanent. If no technique from memory was successful, then a trial and error form of behavior will appear, and solution of the problem may be quite a long

time coming.

As long as the complete memory of the modified Homeostat is saved, say stored on magnetic tape, the system can be stopped and started again at a later time with no loss in operating ability. Even some select portion of memory could be stored on tape and the rest destroyed. If this were the case, then the operation would be somewhat slowed, but the most frequently used A matrices would be kept, and less frequently used A matrices might result as fast from trial and error as from memory search. Again, this is somewhat analogous to a person who is trying to remember the solution to a problem, and less time might actually be required if the person were to "start from scratch" and solve the problem.

Forgetting could be incorporated into the model for memory. The "Success tag" could be made time dependent, with the value of the tag dropping one for each period of x hours or days in which the A matrix was not successful. The A matrices being assigned the condition of permanent memory would not be subject to this time dependence.

One form of learning not shown by the modified Homeostat is that of association. If a new input environment occurs that is similar to an input previously occurring in a different pattern class, the previous adaptations A matrices will never be considered. A form of

association learning could be built into the program by stopping the memory search in the given pattern input class, and switching to the other pattern class storage areas in a random manner. For example, if an input is classed as pattern 2, the pattern 2 memory can be searched through the highest, say, 3 tags. If a successful A matrix has not been located during this search, stop and go to pattern 3 memory and try 5 random A matrices. If this is not successful, go to pattern 4 memory and try 5 random A matrices. If this is not successful go to pattern 1 and try 5 random A matrices. If this is not successful, switch back to pattern 2 memory and continue the detailed search. However, if any of the A matrices from the other pattern classes produced a successful adaptation, then associative learning occurred. This particular A matrix will then be stored in two pattern class memory areas.

Another possibility is that when an A matrix from one class is used to give adaptation for an input from a different class, that A matrix could be tagged and placed in a new category of "generalized adaptations". If a reasonable sized memory of these more generalized A matrices were available, the memory search for any problem could begin in this area.

A point related to the need for allocating the contents of a finite memory is that A matrices with the smallest success tags could be eliminated. This would

mean that over a period of time, the contents of the memory would be A matrices with the largest success tags. The A matrices used the least amount would be erased. This process is analogous to the selective process of the human memory. Only those events which have been heavily reinforced are retained in the accessable memory. Events which occurred without subsequent reinforcement are soon forgotten. The finite space of the memory is used to hold those events which have the most significance, i.e. the ones which have been most useful over some previous period. No one is amazed that he fails to remember an isolated event that occurred some time in the past. Even events with high significance value are forgotten over a period of time where their use is not required.

As mentioned in Chapter V, the modified Homeostat can provide a possible explanation for effects of heredity or instinctive behavior patterns. If the memory originally were stocked with stable A matrices, then adaptation to certain forms of input could be greatly speeded up. The inherited A matrices could be quite non-specific in type of inputs for which they produce adaptation, thus providing a large base of adaptive behavior from which to proceed.

CHAPTER VII

FINAL EXAMPLE AND CONCLUSIONS

A Homeostat Example Based on a Dynamic Model of an Ecological System

As a final example of the behavior of the computer model of the Homeostat, consider the dynamic model of an ecological system.[7] Animal ecology is concerned with the interrelations between animals and their environment. In this case the environmental factors of interest are availability of food for carnivorous species, and co-existence between competing species. With regard to a particular environment, a certain species may adjust to the total environment; it may seek another environment; or it may become extinct. If a species adapts to the environment, this means it can compete or co-operate with other species for food and other needs.

Consider the interdependence of two animal species, one of which serves as the food supply for the other. In isolation, the species serving as the food supply can be

---

[7]John G. Kemeny, and J. Laurie Snell, Mathematical Models in the Social Sciences (Boston: Ginn and Company, 1962), Chapter III.

assumed to increase at a rate proportional to the total population. Alternately, in isolation, the other species can be assumed to decrease at a rate proportional to its total population, since in isolation its food supply has been cut off. When placed together the two species can be assumed to change at a rate proportional to the product of the individual populations. Mathematically, this can be stated in the form of two equations.

$$(1)\qquad \begin{aligned} \frac{dx}{dt} &= ax - bxy \\ \frac{dy}{dt} &= -cy + dxy \end{aligned} \qquad a,b,c,d > 0$$

Here, x represents the populations of the food source, and y represents the population of the other species.

Next consider the interdependence of two animal species which are in competition with each other. In this case each species is assumed to increase at a rate proportional to the population, and to decrease at a rate proportional to the product of the individual populations. The equations describing this interdependence are known as Volterra's competition equations.

$$(2)\qquad \begin{aligned} \frac{dx}{dt} &= ax - bxy \\ \frac{dy}{ct} &= cy - dxy \end{aligned} \qquad a,b,c,d > 0$$

Both sets of equations presented are similar in

form, and are both included in the general set of equations.

$$(3) \qquad \begin{aligned} \frac{dx}{dt} &= a_{11}\, x + a_{12}\, xy \\ \frac{dy}{dt} &= a_{21}\, xy + a_{22}\, y \end{aligned}$$

where the a_{ij} terms can assume arbitrary signs. In additon, this formulation includes a number of other ecological configurations.

Given some initial values for the populations of the two species, we are interested in which configurations lead to conditions of equilibrium for the species. This problem, or ecological system, is applicable to investigation using the computer model of the Homeostat. In other words, we would like to make an ultrastable system using the equations 3 above. This requires a change from the present Homeostat model which uses a set of linear equations. The equations in 3 are nonlinear, and may have more than one equilibrium point. In addition, it is generally quite difficult to determine the stability of a set of nonlinear differential equations.

We can obtain a general idea of the behavior of equations 3 by locating the equilibrium, or singular points, and investigating the solutions of the original equations linearized about each singular point. Taking equations 3, we can form a single equation

$$(4) \qquad \frac{dy}{dx} = \frac{a_{21}\, xy + a_{22}\, y}{a_{11}\, x + a_{12}\, xy}$$

which has its singular points at values of x and y which simultaneously make the numerator and denominator expressions equal to zero. Thus, the singular points are found by solving the equations

$$(5) \qquad \begin{aligned} a_{21}\, xy + a_{22}\, y &= 0 \\ a_{11}\, x + a_{12}\, xy &= 0 \end{aligned}$$

giving solutions of $x = y = 0$, and $x = -a_{22}/a_{21}$, $x=-a_{11}/a_{12}$. Now we will need to investigate each singular point individually.

Near the singular point at $x = y = 0$, we can linearize equations 3 to give

$$(6) \qquad \begin{aligned} \dot{x} &= a_{11}\, x & \dot{x} &= \frac{dx}{dt} \\ \dot{y} &= a_{22} y & \dot{y} &= \frac{dy}{dt} \end{aligned}$$

This linear set of equations will be stable if a_{11} and a_{22} are both negative, and unstable if either a_{11} or a_{22} or both a_{11} and a_{22} are positive. Thus, the stability of the singular point at the origin, and consequently, some small region near the origin, can be determined by considering the signs of only a_{11} and a_{22}.

Near the singular point at $x = -a_{22}/a_{21}$, $y = -a_{11}/a_{12}$ we must make a change of variable in order to allow proper linearization. Let $x_1 = x + a_{22}/a_{21}$ and $y_1 = y + a_{11}/a_{12}$. Substituting these into equations 3 and keeping only the linear terms, we are left with a set of equations

$$(7) \qquad \dot{x}_1 = -\frac{a_{12}}{a_{21}} a_{22} y_1$$

$$\dot{y}_1 = -\frac{a_{21}}{a_{12}} a_{11} x_1$$

which can be given in matrix form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{21}} a_{22} \\ -\frac{a_{21}}{a_{12}} a_{11} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

The stability of this set of equations can be determined by finding the characteristic roots of the coefficient matrix.

Using $\text{DET} \left[A - \lambda I \right] = 0$, we have

$$\begin{vmatrix} -\lambda & \frac{a_{12}}{a_{21}} a_{22} \\ \frac{a_{21}}{a_{12}} a_{11} & -\lambda \end{vmatrix} = 0$$

or $\lambda^2 - a_{11}a_{22} = 0$. Thus, $\lambda = \pm \sqrt{a_{11}a_{22}}$. For a_{11} and a_{22} of the same sign, the characteristic roots will be real with opposite signs, and therefore unstable. For a_{11} and a_{22} opposite in sign, the roots are complex conjugates, and a stable oscillation will result.

Therefore, the most interesting behavior will occur in the vicinity of this second singular point. Here, it will be possible to have a periodic form of behavior with

the populations of the two species showing cyclic variations, while still remaining within some finite limits. For the Homeostat example, we can let x(o) and y(o) represent the initial numbers of the species. Next we must define a maximum number for each species, and call those numbers xCRIT and yCRIT. The system would then be considered adapted to the environment if x and y remain less than xCRIT and yCRIT respectively.

A number of computer runs were necessary to obtain the following five stable system configurations. Most of the runs resulted in unstable systems which failed to adapt in a fixed length of time. Table 8 gives the results of the five successful runs, showing the system configurations and specifying the type of behavior shown.

TABLE 8

RESULTS OF SIMULATION OF ECOLOGICAL SYSTEM

Run Number	A_{11}	A_{12}	A_{21}	A_{22}	Form of Behavior
1	1	-4	7	2	Cyclic Variation
2	-4	1	9	-9	Extinction of Both Species
3	-7	1	2	-6	Extinction of Both Species
4	-3	7	-8	4	Cyclic Variation
5	-1	3	-7	3	Cyclic Variation

CONCLUSIONS

One of the goals of this paper was to describe a procedure to simulate the Homeostat on a large scale general purpose digital computer. This goal was fully accomplished, with the program being given in appendix A, and examples of the behavior of the computer simulation given in Chapter IV. Additionally, examples of Homeostat behavior were given for number of variables equal to seven and ten. This work represented a demonstration of certain statements made by Ashby, which he was unable to verify experimentally.

The second goal of this paper was to modify the computer simulation of the Homeostat to incorporate the functions of memory and learning into its behavior. The results given in Chapter VI show that the addition of the memory function significantly alters the behavior of the basic Homeostat. A quite unexpected result was the almost total ability of the memory to produce adaptation to a wide range of inputs. In Table 3 of Chapter VI this result is shown. Out of 20 inputs only 3 adaptations by trial and error were required to produce 13 adaptations by use of the memory function. Results from Table 5 of Chapter VI show that following a sequence of 20 inputs used as a training sequence, the probability of an adaptation from memory was 90 per cent. This figure of 90

per cent was computed on the basis of a random set of data, and may vary considerably for different data sets. However, on subsequent computer runs, the number of adaptations from memory was always near this value.

One of the main factors given for not accepting the concept of ultrastability as a valid model of the nervous system is that the time to adapt using ultrastability is far excessive to that actually shown. However, the results obtained show that a very small memory of successful adaptations produced by trial and error can result in a large number of adaptations of future inputs using this memory. Thus after some training sequence, the modified Homeostat would seem to be a much more acceptable model since it greatly reduces the adaptation time required for new inputs.

Because of the unexpected great success of the memory addition, the effect of adding the learning function was quite unspectacular. The main reason for this is that with the small number of A matrices stored in memory, the special search technique used is too restricted. Even with a computer run using 50 input patterns only 4 A matrices were stored in memory. Thus, most any reasonable number of inputs will produce a scant number of A matrices for the memory. Here, a reasonable number of inputs is defined by the length of computer time required to process the inputs. The results obtained for other computer runs

were almost random. In some cases the learning scheme reduced the search time, but in other cases it lengthened it. The results of Table 7 in Chapter VI show one run where the effect of the learning scheme was such that the overall search time was significantly reduced.

The final example of the simulation of an ecological system was included to show the general utility of an ultrastable system model. The purpose of the simulation was to obtain a number of stable ecological systems in response to varied input patterns. The results showed that the ultrastable system model was useful for predicting stable system configurations. Some additional comments on other uses for ultrastable system models are given in the next section.

TOPICS FOR FURTHER STUDY

During the course of this work a number of topics arose which were quite interesting, and appear to be worthy of further study. However, the pursuit of these topics is beyond the scope of this work. Therefore, these topics are collected here for the purpose of providing apparently worthwhile ideas for extensions and/or modifications to this work.

An elementary extension of the size of the basic Homeostat simulation could easily be made, and some examples run to show the actual behavior for a Homeostat with

several hundred variables. An extensive effort could be devoted to a more comprehensive pattern description and recognition system. A possible view here could be to classify a number of human learned responses for input to the Homeostat. A study could also be made to determine the number of input patterns required to provide a meaningful simulation of a particular system.

Within the framework of the concept of ultrastability there might be some other search scheme for A matrices besides the pure random search. Presently, an A matrix is modified on a row basis relative to the particular variable that exceeded its critical state. It might be possible to obtain a faster rate of convergence by modifying the whole A matrix; or possibly by modifying only a single element of the A matrix. Here, a wide variety of methods are available for analysis. With regard to the learning techniques, a number of different learning schemes could be tried, or using the method proposed, other search techniques could be investigated.

A very interesting topic which would require a detailed analysis is the explanation of the result from page 52 that 3 A matrices produced adaptation to 18 inputs. During one particular computer run, a single A matrix was used 31 times. The reason for a particular A matrix producing adaptation for a wide variety of inputs is not clear. Analysis of this phenomenon could lead to a

technique for constructing or selecting A matrices that are universal, or at least quite general in use.

A number of topics were suggested in the section relating the Homeostat operation to operation of the human brain in Chapter VI. The effect of loss of memory sections could be studied; or different models for forgetting could be tried and tested. The addition of learning by association would be a very worthwhile addition. Also the effects of heredity could be studied by initially stocking the memory of the modified Homeostat with different types and numbers of A matrices.

A final area of further study is the use of ultrastability in the modelling of general systems. In other words what type of systems are amenable to simulation using the principle of the Homeostat. One possibility here might be political systems in countries with dictatorships. A revolution could be considered as a random change in the system due to a certain systems variable having exceeded its critical state. A wide variety of other systems can also be studied using the concept of ultrastability.

REFERENCES

Books

Anderson, A. R. Minds and Machines, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964.

Ashby, W. Ross. Design For a Brain, John Wiley and Sons, New York, 1960.

_____. An Introduction to Cybernetics, Science Editions, John Wiley and Sons, New York, 1963.

Bell, D. A. Intelligent Machines, Blaisdell Publishing Company, New York, 1962.

Cherry, C., Editor. Information Theory, Fourth London Symposium, Butterworths, Inc., Washington D. C., 1961.

Cunningham, W. J. Introduction to Non-linear Analysis, McGraw-Hill Book Company, Inc., New York, 1958.

Feigenbaum, E. A. and Feldman, J., Editors. Computers and Thought, McGraw-Hill Book Company, New York, 1963.

Formby, John. An Introduction to the Mathematical Formulation of Self-Organizing Systems, D. Van Nostrand Company, Inc., Princeton, New Jersey, 1965.

George, F. H. Automation, Cybernetics and Society, Philosophical Library, New York, 1959.

Glushkov, V. M. Introduction to Cybernetics, Academic Press, New York, 1966.

Hill, J. D., McMurtry, G. J., and Fu, K. S. A Computer-Simulated On-Line Experiment in Learning Control Systems. AFIPS Conference Proceedings, Vol. 25, 1964 Spring Joint Computer Conference, Spartan Books, Inc., Baltimore, Maryland, 1964.

Hilton, A. M. Logic, Computing Machines, and Automation, Spartan Books, Washington, D. C., 1963.

Kemeny, J. G., and Snell, J. L. Mathematical Models in the Social Sciences, Ginn and Company, Boston, 1962.

Kemeny, J. G., Snell, J. L., and Thompson, G. L. Finite Mathematics, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1957.

Lapidus, L. Digital Computation for Chemical Engineers, McGraw-Hill Book Company, New York, 1962.

McCracken, D. D., and Dorn, W. S. Numerical Methods and FORTRAN Programming, John Wiley & Sons, Inc., New York, 1964.

Mishkin, E., and Braun, L. Adaptive Control Systems, McGraw-Hill Book Company, 1961.

Muses, C. A., Editor. Aspects of the Theory of Artificial Intelligence, Plenum Press, New York, 1962.

Naylor, T. H., et al. Computer Simulation Techniques, John Wiley and Sons, Inc., New York, 1966.

Nilsson, N. J. Learning Machines, McGraw-Hill Book Company, New York, 1965.

Pask, G. An Approach to Cybernetics, Hutchinson and Company, London, 1961.

Pedelty, M. J. An Approach to Machine Intelligence, Spartan Books, Washington, D. C., 1963.

Sayre, K. M., and Crosson, F. J., Editors. The Modeling of Mind, University of Notre Dame Press, Notre Dame, Indiana, 1963.

Singh, J. Great Ideas in Information Theory, Language and Cybernetics, Dover Publications, Inc., New York, 1966.

Tou, J. T. Modern Control Theory, McGraw-Hill Book Company, New York, 1964.

Tsien, H. S. Engineering Cybernetics, McGraw-Hill Book Company, New York, 1954.

Walter, W. Grey. The Living Brain, W. W. Norton and Company, New York, 1953.

Wiener, Norbert. Cybernetics, The Technology Press, John Wiley and Sons, New York, 1948.

Woolridge, Dean E. The Machinery of the Brain, McGraw-Hill Book Company, New York, 1963.

Yovits, M. C., and Cameron, S., Editors. Self-Organizing Systems, Pergamon Press, New York, 1960.

Yovits, M. C., Jocobi, G. T., and Goldstein, G. D. Self-Organizing Systems 1962, Spartan Books, Washington, D. C., 1962.

Zadeh, L. A., and Desoer, C. A. Linear System Theory, McGraw-Hill Book Company, New York, 1963.

Articles

Atzenbeck, C. R., and Hampel, D. "Neural, Threshold, Majority, and Boolean Logic Techniques", RCA Publication PE-233, Life Sciences, pages 31-35, 1965.

Chichinaldze, V. K. "On the Dynamics of Some Learning and Self-Learning Processers", Automatic and Remote Control Proceedings of the International Federation of Automatic Control, 1963, Butterworth and Company, Ltd., London, 1964.

Hawkins, J. K. "Self Organizing Systems - A Review and Commentary", Proc. IRE, vol. 49, number 1, 1961, pages 31-48.

Hormann, A. M. "Gaku: An Artificial Student", Behavioral Science, vol. 10, number 1, 1965, pages 88-100.

Lynn, J. W., and Goldrigh, J. M. "Analytical Techniques in Brain Modelling", Bionics Symposium, 1966, 3-5 May 1966, Wright-Patterson Air Force Base, Ohio.

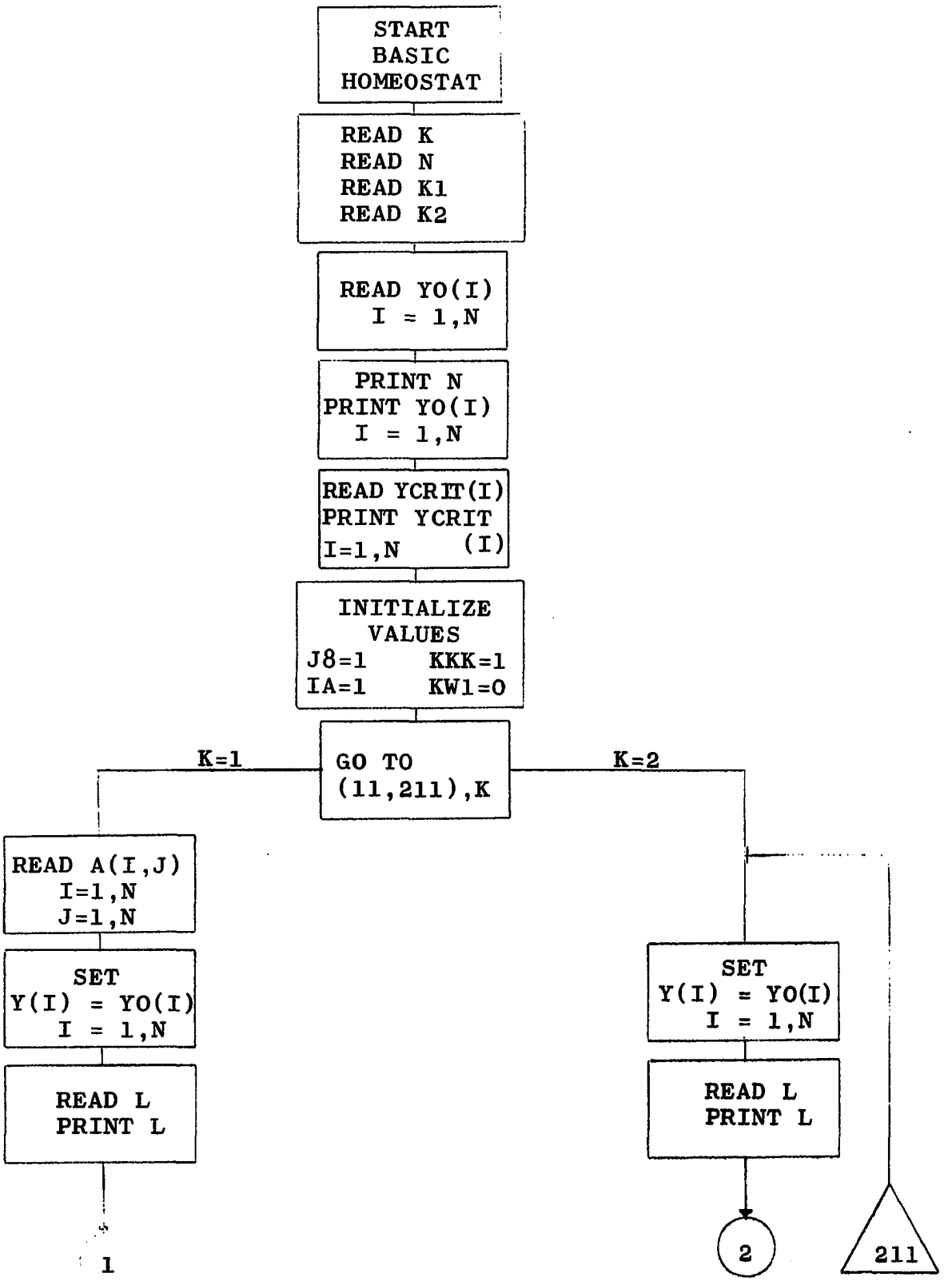
Minsky, Marvin. "Steps Toward Artificial Intelligence", Proc. IRE, vol. 49, number 1, 1961, pages 8-30.

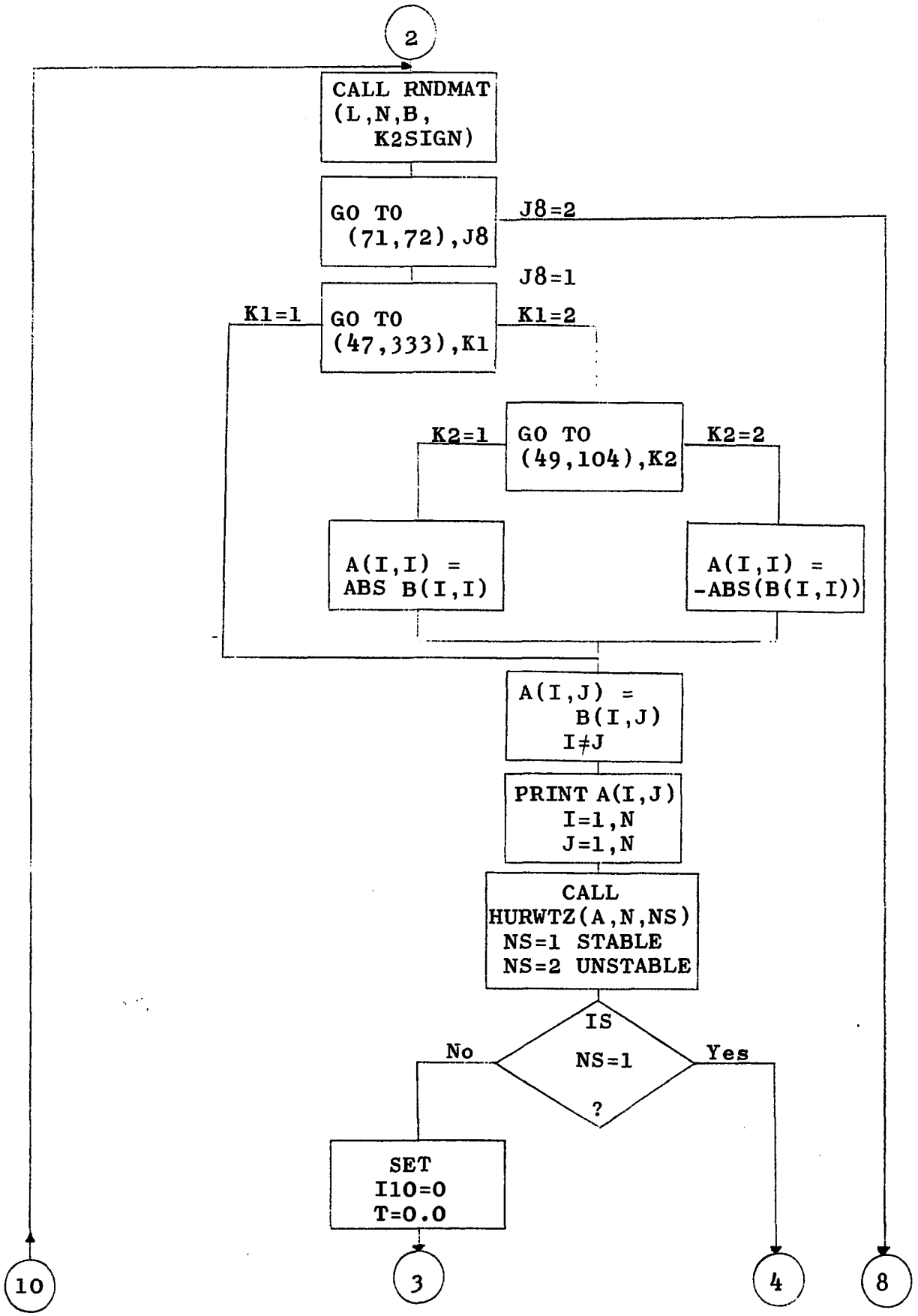
Sklansky, J. "Adaptation Theory", RCA Publication PE-232, 1965.

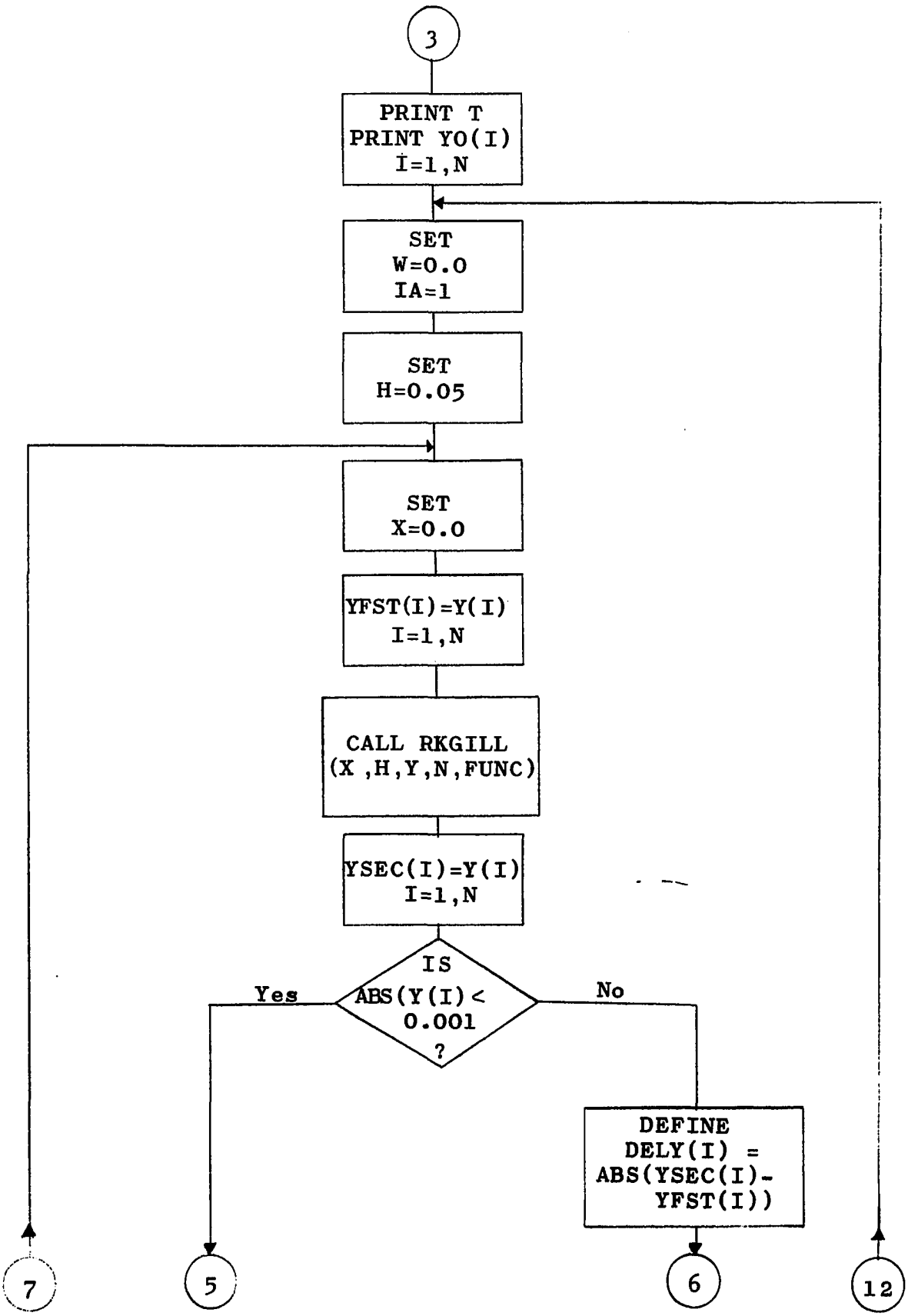
APPENDIX A

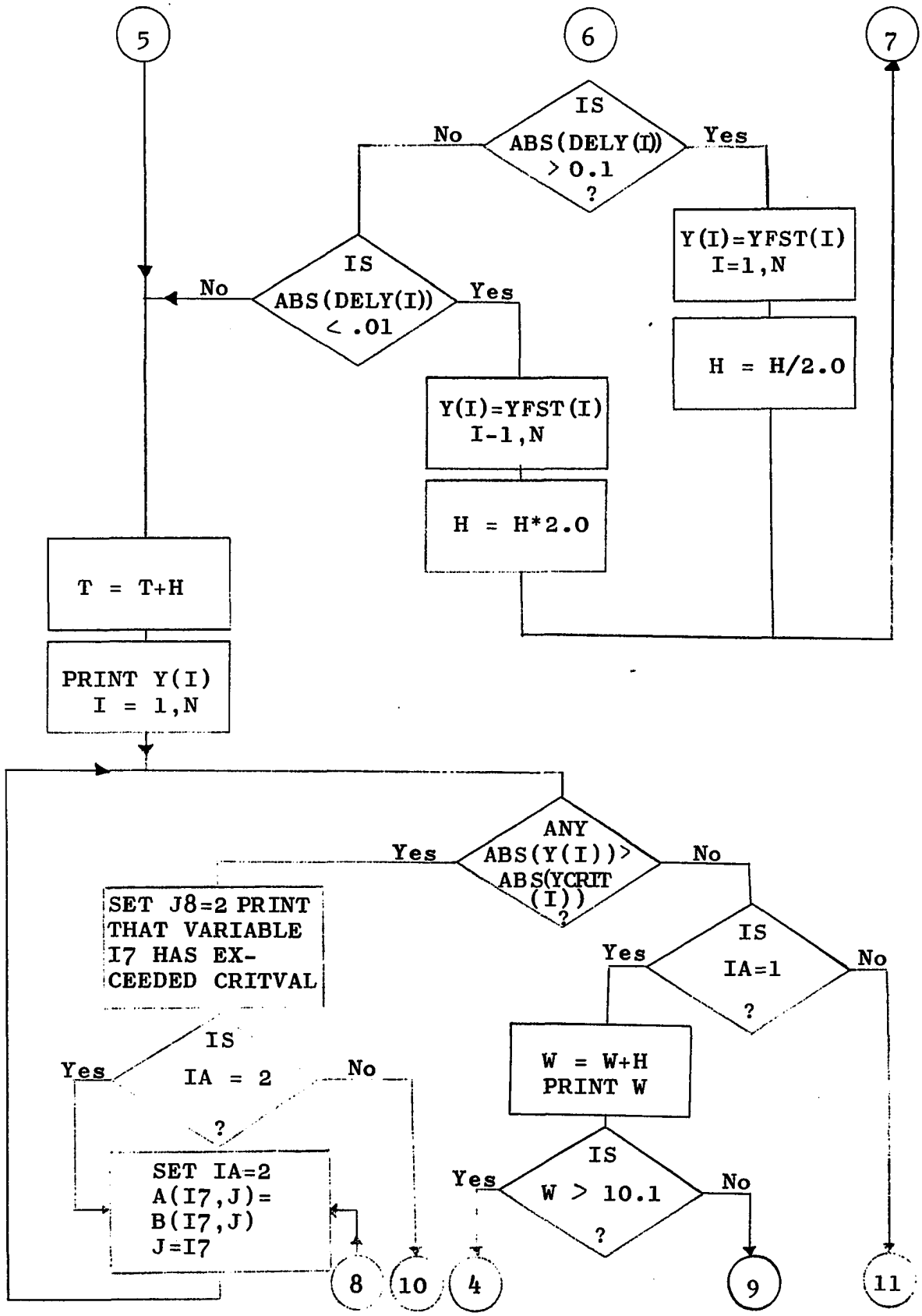
FLOW CHART AND LISTING OF THE COMPUTER PROGRAM USED TO SIMULATE THE BASIC HOMEOSTAT

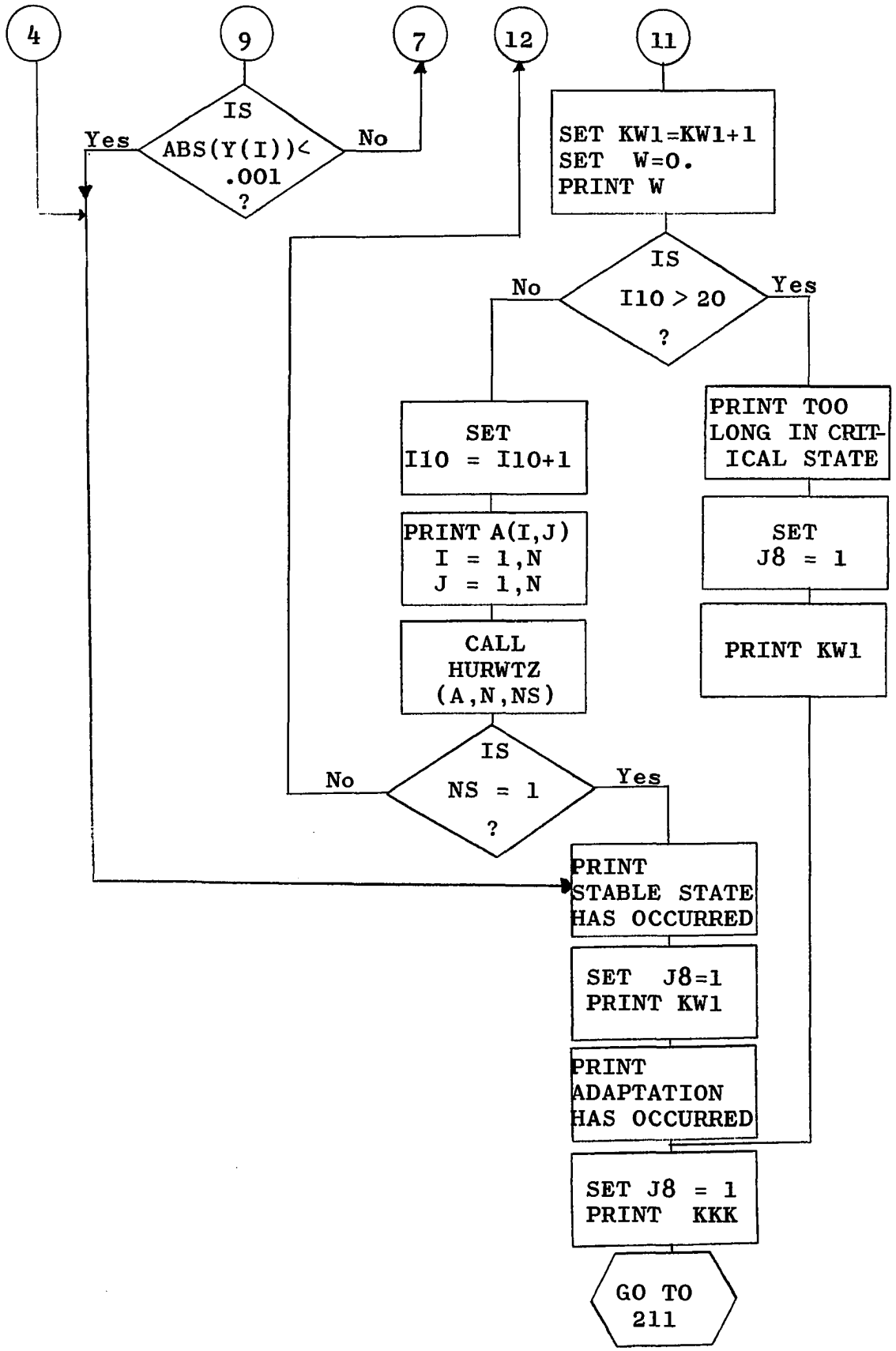
FLOW CHART FOR BASIC HOMEOSTAT COMPUTER MODEL




2
CALL RNDMAT (L,N,B, K2SIGN)
GO TO (71,72),J8
J8=2
J8=1
K1=1
GO TO (47,333),K1
K1=2
K2=1
GO TO (49,104),K2
K2=2
A(I,I) = ABS B(I,I)
A(I,I) = -ABS(B(I,I))
A(I,J) = B(I,J) I≠J
PRINT A(I,J) I=1,N J=1,N
CALL HURWTZ(A,N,NS) NS=1 STABLE NS=2 UNSTABLE
IS NS=1 ?
No
Yes
SET I10=0 T=0.0
10
3
4
8


3
PRINT T
PRINT YO(I)
I=1,N
SET
W=0.0
IA=1
SET
H=0.05
SET
X=0.0
YFST(I)=Y(I)
I=1,N
CALL RKGILL
(X ,H,Y,N,FUNC)
YSEC(I)=Y(I)
I=1,N
IS
ABS(Y(I)<
0.001
?
Yes
No
DEFINE
DELY(I) =
ABS(YSEC(I)-
YFST(I))
7
5
6
12


5
6
7
IS ABS(DELY(I)) > 0.1 ?
No
Yes
Y(I)=YFST(I) I=1,N
H = H/2.0
IS ABS(DELY(I)) < .01
No
Yes
Y(I)=YFST(I) I-1,N
H = H*2.0
T = T+H
PRINT Y(I) I = 1,N
ANY ABS(Y(I))> ABS(YCRIT (I))
Yes
No
SET J8=2 PRINT THAT VARIABLE I7 HAS EX-CEEDED CRITVAL
IS IA=1 ?
Yes
No
W = W+H PRINT W
IS W > 10.1 ?
Yes
No
IS IA = 2 ?
Yes
No
SET IA=2 A(I7,J)= B(I7,J) J=I7
8
10
4
9
11


4
9
7
12
11
IS
ABS(Y(I))<
.001
?
Yes
No
SET KW1=KW1+1
SET W=0.
PRINT W
IS
I10 > 20
?
No
Yes
SET
I10 = I10+1
PRINT TOO
LONG IN CRIT-
ICAL STATE
PRINT A(I,J)
I = 1,N
J = 1,N
SET
J8 = 1
CALL
HURWTZ
(A,N,NS)
PRINT KW1
IS
NS = 1
?
No
Yes
PRINT
STABLE STATE
HAS OCCURRED
SET J8=1
PRINT KW1
PRINT
ADAPTATION
HAS OCCURRED
SET J8 = 1
PRINT KKK
GO TO
211

```
C      BASIC HOMEOSTAT   IBM 1410 SIMULATION FORTRAN II
       DIMENSION A(10,10),B(10,10),Y(10),YO(10),YCRIT(10)
       DIMENSION YFST(10),YSEC(10),DELY(10)
       COMMON N,A
       READ 1, K
       READ1, N
       READ 1, K1
       READ 1, K2
     1 FORMAT(I3)
   110 DO 2 I=1,N
     2 READ 3, YO(I)
     3 FORMAT(F10.0)
       PRINT 4, N
     4 FORMAT(1X,23H THE NUMBER OF UNITS IS,13,//)
       PRINT 5
     5 FORMAT(1X,36H THE INITIAL METER DISPLACEMENTS ARE, //)
       PRINT 6, (YO(I),I=1,N)
     6 FORMAT(1X,10F8.2,///)
       DO 101 I=1,N
   101 READ 3,YCRIT(I)
       PRINT 201
   201 FORMAT(///,24H THE CRITICAL STATES ARE,//)
       PRINT 6,(YCRIT(I),I=1,N)
       J8=1
       KKK=1
       GO TO(11,211),K
    11 DO 8 I=1,N
     8 READ 21, (A(I,J),J=1,N)
    21 FORMAT(10F8.0)
       DO 1000 I=1,N
  1000 Y(I)=YO(I)
       READ 210, L
       PRINT 105, L
   105 FORMAT(1X,3H L=,I10,//)
       GO TO 99
   211 DO 7 I=1,N
     7 Y(I)=YO(I)
    12 READ 210,L
   210 FORMAT(I6)
       PRINT 105, L
    13 CALL RNDMAT(L,N,B,K2SIGN)
       IA=1
    36 GO TO(71,72),J8
    71 DO 41 I2=1,N
       DO 41 J2=1,N
       IF(I2-J2)49,48,49
    48 GO TO(47,333),K1
    47 READ 45,A(I2,I2)
    45 FORMAT(F10.0)
       GO TO 41
   333 GO TO(49,104),K2
   104 B(I2,J2)=-(ABS(B(I2,J2)))
```

```
 49 A(I2,J2)=B(I2,J2)
 41 CONTINUE
 99 PRINT 433
433 FORMAT(///,16H THE A MATRIX IS,//)
    DO 131 I=1,N
    PRINT 132,(A(I,J),J=1,N)
132 FORMAT(1X,10F8.2)
131 CONTINUE
    CALL HURWTZ(A,N,NS)
    IF(NS-1)81,81,396
396 I10=1
    T=0.0
    PRINT 450,T
450 FORMAT(///,18H BEGIN TRIAL AT T=,F10.8)
    PRINT 451
451 FORMAT(///,14H THE YO(I) ARE)
    PRINT 6, (YO(I),I=1,N)
    KW1=0
 76 W=0.0
    IA=1
    H=0.05
 62 X=0.0
    DO 635 I=1,N
635 YFST(I)=Y(I)
    CALL RKGILL(X,H,Y,N,FUNC)
    DO 636 I=1,N
636 YSEC(I)=Y(I)
    DO 480 I=1,N
    IF(ABS(Y(I))-0.001)480,480,481
480 CONTINUE
    GO TO 469
481 DO 470 I=1,N
    DELY(I)=ABS(YSEC(I))-ABS(YFST(I))
    IF(ABS(DELY(I))-0.1)470,470,463
470 CONTINUE
    GO TO 462
463 H=H/2.0
467 DO 466 I=1,N
466 Y(I)=YFST(I)
    GO TO 62
462 DO 465 I=1,N
    IF(ABS(DELY(I))-0.01)465,469,469
465 CONTINUE
    H=H*2.0
    GO TO 467
469 T=T+H
    PRINT 452,T
452 FORMAT(1X,6H AT T=,F20.8,14H  THE Y(I) ARE,//)
    PRINT 6,(Y(I),I=1,N)
    DO 52 I7=1,N
    IF(ABS(YCRIT(I7))-ABS(Y(I7)))54,54,52
 52 CONTINUE
```

```
485 GO TO(631,610),IA
631 W=W+H
    PRINT 499,W
499 FORMAT(1X,3H W=,F20.8,//)
    GO TO 61
 54 PRINT 102, I7
102 FORMAT(1X,3H X(,I2,33H) HAS EXCEEDED ITS CRITICAL
      STATE,//)
    J8=2
    GO TO(13,72),IA
610 KW1=KW1+1
    W=0.0
    PRINT 499,W
    IF(I10-20)91,92,92
 92 PRINT 93
 93 FORMAT(1X,20H TOO LONG IN CRIT ST)
    J8=1
    PRINT 606, KW1
606 FORMAT(1X,I5,24H   STEP CHANGES OCCURRED,//)
    GO TO 103
 91 I10=I10+1
    GO TO 611
 72 DO 73 I=1,N
    IF(I7-I)74,75,74
 75 GO TO 73
 74 A(I7,I)=B(I7,I)
 73 CONTINUE
    IA=2
    GO TO 52
611 PRINT 433
    DO 151 I=1,N
    PRINT 132,(A(I,J),J=1,N)
151 CONTINUE
395 CALL HURWTZ(A,N,NS)
    IF(NS-1)81,81,76
 61 IF(W-10.1)82,82,81
 82 DO 473 I=1,N
    IF(ABS(Y(I))-0.001)473,473,472
473 CONTINUE
    GO TO 81
472 GO TO 62
 81 PRINT 83
 83 FORMAT(1X,26H STABLE STATE HAS OCCURRED)
    PRINT 606, KW1
780 PRINT 781
781 FORMAT(1X,24H ADAPTATION HAS OCCURRED)
103 PRINT 106,KKK
106 FORMAT(11H RUN NUMBER,I6)
    J8=1
    KKK=KKK+1
    GO TO 211
    END
```

APPENDIX B

LISTING OF FORTRAN SUBROUTINES USED IN ALL COMPUTER PROGRAMS SIMULATING THE HOMEOSTAT

```
      SUBROUTINE RNDMAT(L,N,B,K2SIGN)
      DIMENSION B(5,5)
   13 DO 31 I1=1,N
      DO 31 J1=1,N
  301 F=RANDOM(L)
      F1=10.0*F
      M1=F1
      IF(M1)302,301,302
  302 F2=100.0*F
      M2=F2
      M3=M2/2
      F3=M3
      F4=M2
      F5=F4/2.0
      IF(F3-F5)32,33,34
   32 B(I1,J1)=-M1
      GO TO 31
   33 B(I1,J1)=M1
      GO TO 31
   34 PRINT 35
   35 FORMAT(1X,14H MACHINE ERROR)
      STOP
   31 CONTINUE
      GO TO (10,11),K2SIGN
   11 DO 12 I=1,N
   12 B(I,I)=-(ABS(B(I,I)))
   10 RETURN
      END

      SUBROUTINE RKGILL(X,H,Y,N,FUNC)
      DIMENSION Y(5),YK(5),Q(5),YP(5)
      DIMENSION A(5,5)
      EQUIVALENCE (YK,YP)
      R=H/6.0
      Z=X+0.5*H
      CALL FUNC(X,Y,YP)
      DO 1 I=1,N
C     COMPUTE K1,J1,Q1
      YK(I)=H*YP(I)
      Y(I)=Y(I)+0.5*YK(I)
    1 Q(I)=YK(I)
      CALL FUNC(Z,Y,YP)
      DO 2 I=1,N
C     COMPUTE K2,J2,Q2
      YK(I)=H*YP(I)
      Y(I)=Y(I)+.2928932188134524760 0*(YK(I)-Q(I))
    2 Q(I)=.5857864376269049520 0 * YK(I) + .1213203435596
        2572 *Q(I)
      CALL FUNC(Z,Y,YP)
      DO 3 I=1,N
C     COMPUTE K3,J3,Q3
```

```
      YK(I)=H*YP(I)
      Y(I)=Y(I)+ 1.7071067811865475 24 * (YK(I) - Q(I))
    3 Q(I)= 3.4142135623730950 48 * YK(I)  -4.1213203435596
        42572 * Q(I)
      X=X+H
      CALL FUNC(X,Y,YP)
      DO 4 I=1,N
C     COMPUTE J4
    4 Y(I)=Y(I)+R*YP(I)-Q(I)/3.0
      RETURN
      END

      SUBROUTINE FUNC(X,Y,YP)
      DIMENSION A(5,5),Y(5),YP(5)
      COMMON N,A
      DO 1 I=1,N
      YP(I)=0.0
      DO 1 J=1,N
    1 YP(I)=YP(I)+A(I,J)*Y(J)
      RETURN
      END

      SUBROUTINE HURWTZ(A,N,NS)
      DIMENSION A(5,5),ACHAR(10)
      CALL CHAREQ(A,N,ACHAR)
      PRINT 20, (ACHAR(I),I=1,N)
   20 FORMAT(1X,5E20.8)
      DO 60 I=1,N
      IF(ACHAR(I))2,3,4
   60 CONTINUE
    4 D2=(ACHAR(1)*ACHAR(2)-ACHAR(3))
      IF(D2)2,3,5
    5 E1=ACHAR(3)*D2
      E2=ACHAR(1)*(ACHAR(1)*ACHAR(4)-ACHAR(5))
      D3=E1-E2
      IF(D3)2,3,6
    6 F1=ACHAR(4)*D3
      G1=ACHAR(5)*D2
      G2=ACHAR(1)*(ACHAR(1)*ACHAR(6)-ACHAR(7))
      F2=-(G1-G2)*ACHAR(2)
      H1=ACHAR(5)*(ACHAR(1)*ACHAR(4)-ACHAR(5))
      H2=ACHAR(3)*(ACHAR(1)*ACHAR(6)-ACHAR(7))
      F3=H1-H2
      D4=F1-F2+F3
      IF(D4)2,3,7
    7 PRINT 8
      NS=1
    8 FORMAT(1X,21H THE SYSTEM IS STABLE)
      GO TO 50
    2 PRINT 9
      NS=2
    9 FORMAT(1X,23H THE SYSTEM IS UNSTABLE)
```

```
      GO TO 50
    3 PRINT 10
      NS=3
   10 FORMAT(1X,6H DET=0)
   50 RETURN
      END
      SUBROUTINE CHAREQ(A,N,ACHAR)
      DIMENSION A(5,5),C(5,5),D(5,5),TR(5),ACHAR(10)
      TR(1)=0.0
      DO 1 I=1,N
    1 TR(1)=TR(1)+A(I,I)
   20 FORMAT(1X,5E20.8)
      DO 2 I=1,N
      DO 2 J=1,N
    2 D(I,J)=A(I,J)
      DO 3 I=2,N
      CALL MATM10(A,D,C,N)
      TR(I)=0.0
      DO 4 J=1,N
    4 TR(I)=TR(I)+C(J,J)
      DO 5 I2=1,N
      DO 5 J2=1,N
    5 D(I2,J2)=C(I2,J2)
    3 CONTINUE
      ACHAR(1)=-TR(1)
      PRINT 20, ACHAR(1)
      DO 10 K=2,N
      SUM=0.0
      L=K-1
      DO 11 I=1,L
      M=K-I
   11 SUM=SUM+ACHAR(I)*TR(M)
      AK=K
      ACHAR(K)=-(1.0/AK)*(SUM+TR(K))
   10 PRINT 20, ACHAR(K)
      NN=N+1
      DO 30 I=NN,10
   30 ACHAR(I)=0.0
      RETURN
      END
      SUBROUTINE MATM10(A,D,C,N)
      DIMENSION A(5,5),D(5,5),C(5,5)
C     C=A*D
      DO 1 J=1,N
      DO 1 I=1,N
      T = 0.
      DO 2 K=1,N
    2 T=T+A(I,K)*D(K,J)
    1 C(I,J) = T
      RETURN
      END
```

APPENDIX C

LISTING OF THE COMPUTER PROGRAM TO SIMULATE THE HOMEOSTAT WITH INPUT PATTERN RECOGNITION CAPABILITY

```
C      HOMEOSTAT WITH PATTERN CLASSIFICATION   IBM 1410
         SIMULATION
       DIMENSION A(5,5),B(5,5),YO(5),Y(5),YCRIT(5),YFST(5),
         YSEC(5)
       DIMENSION DELY(5),A1(1,2),A2(1,2)
       COMMON N,A
       READ1, K
       READ1, N
       READ 1, K1
       READ 1, K2
     1 FORMAT(I3)
       KALT=0
       KTRLCT=0
       KSW=1
       READ 161, NVA
   161 FORMAT(I3)
       MVA=NVA+1
       GO TO (162,163,163)MVA
   163 READ 165, IJ1,IJ2
   165 FORMAT(2I3)
       GO TO (164,166),NVA
   166 READ 165, IJ3,IJ4
   164 READ 161, KALT
       TO TO (162,168),KALT
   168 READ 265, A1(1,1),A2(1,1)
   265 FORMAT(2F4.0)
       GO TO (162,170),NVA
   170 READ 265, A1(1,2),A2(1,2)
       GO TO 162
   162 READ 161, NVR
       MVR=NVR+1
       GO TO (171,172,172),MVR
   172 READ 165, KY1,KY2
       GO TO (173,174),NVR
   174 READ 165,KY3,KY4
   173 READ 161, KREL1
       GO TO (176,175),NVR
   175 READ 161, KREL2
   176 READ 165, KVF1,KVF2
       READ 165, KTRIAL
       IF(KVF2)171,171,332
   332 READ 161, KTWOV
       GO TO 171
   171 IF(NVA)180,180,181
   180 IF(NVR)182,182,183
   182 KPAT=1
       GO TO 190
   183 KPAT=2
       GO TO 190
   181 IF(NVR)184,184,185
   184 KPAT=3
```

```
      GO TO 190
  185 KPAT=4
      GO TO 190
  190 PRINT 315, KPAT
  315 FORMAT(1X,7H KPAT =,I3,//)
      DO 107 I=1,5
      DO 107 J=1,5
      A(I,J)=0.0
  107 B(I,J)=0.0
  110 DO 2 I=1,N
    2 READ 3, YO(I)
    3 FORMAT(F10.0)
      PRINT 4,N
    4 FORMAT(1X,23H THE NUMBER OF UNITS IS,I3,//)
      PRINT 4, N
    5 FORMAT(1X,36H THE INITIAL METER DISPLACEMENTS ARE, //)
      PRINT 6,(YO(I),I=1,N)
    6 FORMAT(1X,10F8.2,///)
      DO 101 I=1,N
  101 READ 3,YCRIT(I)
      PRINT 201
  201 FORMAT(///,24H THE CRITICAL STATES ARE,//)
      PRINT 6,(YCRIT(I),I=1,N)
      J8=1
      KW1A=1
      KKK=1
      KSWCT=0
      KIND=1
      GO TO(11,211),K
   11 DO 8 I=1,N
    8 READ 21, (A(I,J),J=1,N)
   21 FORMAT(10F8.0)
      DO 1000 I=1,N
 1000 Y(I)=YO(I)
      READ 210, L
      GO TO 99
  211 DO 7 I=1,N
    7 Y(I)=YO(I)
   12 READ 210,L
  210 FORMAT(I6)
      PRINT 105, L
  105 FORMAT(1X,3H L=,I10,///)
      K2SIGN=K2
   13 CALL RNDMAT(L,N,B,K2SIGN)
      IA=1
   36 GO TO(71,72),J8
   71 DO 41 I2=1,N
      DO 41 J2=1,N
      IF(I2-J2)49,48,49
   48 GO TO (47,49),K1
   47 READ 45,A(I2,I2)
```

```
   45 FORMAT(F10.0)
      GO TO 41
   49 A(I2,J2)=B(I2,J2)
   41 CONTINUE
      IF(KALT-1)99,99,202
  202 A(IJ1,IJ2)=A1(1,1)
      GO TO (203,204),NVA
  204 A(IJ3,IJ4)=A1(1,2)
  203 KCOUNT=1
   99 PRINT 433
  433 FORMAT(///,16H THE A MATRIX IS,//)
      DO 131 I=1,N
      PRINT 132,(A(I,J),J=1,N)
  132 FORMAT(1X,10F8.2)
  131 CONTINUE
      CALL HURWTZ(A,N,NS)
      IF(NVR)250,250,396
  250 IF(NS-1)81,81,396
  396 I10=1
      T=0.0
      PRINT 450,T
  450 FORMAT(///,18H BEGIN TRIAL AT T=,F10.8)
      PRINT 451
  451 FORMAT(///,14H THE YO(I) ARE)
      PRINT 6, (YO(I),I=1,N)
      KW1=0
   76 W=0.0
      IA=1
      H=0.05
   62 X=0.0
F     FUNC
      DO 635 I=1,N
  635 YFST(I)=Y(I)
      CALL RKGILL(X,H,Y,N,FUNC)
      DO 636 I=1,N
  636 YSEC(I)=Y(I)
      DO 480 I=1,N
      IF(ABS(Y(I))-0.001)480,480,481
  480 CONTINUE
      GO TO 469
  481 DO 470 I=1,N
      DELY(I)=ABS(YSEC(I))-ABS(YFST(I))
      IF(ABS(DELY(I))-0.1)470,470,463
  470 CONTINUE
      GO TO 462
  463 H=H/2.0
  467 DO 466 I=1,N
  466 Y(I)=YFST(I)
      GO TO 62
  462 DO 465 I=1,N
      IF(ABS(DELY(I))-0.01)465,469,469
```

```
465 CONTINUE
    H=H*2.0
    GO TO 467
469 T=T+H
    PRINT 452,T
452 FORMAT(1X,6H AT T=,F20.8,14H  THE Y(I) ARE,//)
    PRINT 6,(Y(I),I=1,N)
    IF(NVR)142,142,212
212 IF(KSWCT-20)276,276,277
277 PRINT 288
288 FORMAT(1X,22H ADAPTATION IMPOSSIBLE)
    GO TO 289
276 IF(KSW-1)141,141,142
141 IF(YFST(KY1))121,122,123
121 IF(YSEC(KY2))126,122,124
126 IND1=1
    KSW=2
    GO TO (125,213),NVR
124 IND1=2
    KSW=2
    GO TO (125,213),NVR
123 IF(YSEC(KY2))124,122,126
122 GO TO 142
213 IF(YFST(KY3))214,215,216
215 KSW=1
    GO TO 142
214 IF(YSEC(KY4))217,215,218
217 IND2=1
    GO TO 230
218 IND2=2
    GO TO 230
216 IF(YSEC(KY4))218,215,217
230 KSW=2
125 KSWCT=KSWCT+1
    IF(NS-1)81,81,142
142 DO 52 I7=1,N
    IF(ABS(YCRIT(I7))-ABS(Y(I7)))54,54,52
 52 CONTINUE
    GO TO (485,275),KIND
275 DO 348 I=1,N
348 Y(I)=YO(I)
    KIND=1
    GO TO 99
485 GO TO(631,610),IA
631 W=W+H
    PRINT 499,W
499 FORMAT(1X,3H W=,F20.8,//)
    GO TO 61
 54 PRINT 102,I7
102 FORMAT(1X,3H X(,I2,33H) HAS EXCEEDED ITS CRITICAL
      STATE,//)
```

```
      J8=2
      GO TO(13,72),IA
  610 KW1=KW1+1
      W=0.0
      PRINT 499,W
      IF(I10-20)91,92,92
   92 PRINT 93
   93 FORMAT(1X,20H TOO LONG IN CRIT ST)
  289 J8=1
      PRINT 606, KW1
  606 FORMAT(1X,I5,24H   STEP CHANGES OCCURRED,//)
      GO TO 103
   91 I10=I10+1
      GO TO 611
   72 DO 73 I=1,N
      IF(I7-I)74,75,74
   75 GO TO 73
   74 A(I7,I)=B(I7,I)
   73 CONTINUE
      IA=2
      GO TO 52
  611 PRINT 433
      DO 151 I=1,N
      PRINT 132,(A(I,J),J=1,N)
  151 CONTINUE
      GO TO 395
  395 CALL HURWTZ(A,N,NS)
      IF(NS-1)81,81,76
   61 IF(W-5.1)82,82,81
   82 DO 473 I=1,N
      IF(ABS(Y(I))-0.001)473,473,472
  473 CONTINUE
      GO TO 81
  472 GO TO 62
   81 PRINT 83
   83 FORMAT(1X,26H STABLE STATE HAS OCCURRED)
      IF(KALT-1)240,192,198
  192 J8=1
      PRINT 606,KW1
      DO 776 I=1,N
  776 Y(I)=YO(I)
      A(IJ1,IJ2)=-A(IJ1,IJ2)
      GO TO (193,194),NVA
  194 A(IJ3,IJ4)=-A(IJ3,IJ4)
      GO TO 193
  198 COUNT=KCOUNT/2
      KCONT1=KCOUNT/2
      COUNT1=KCONT1
      COUNT2=COUNT/2.0
      KVAR=COUNT1-COUNT2
      IF(KVAR)200,199,200
```

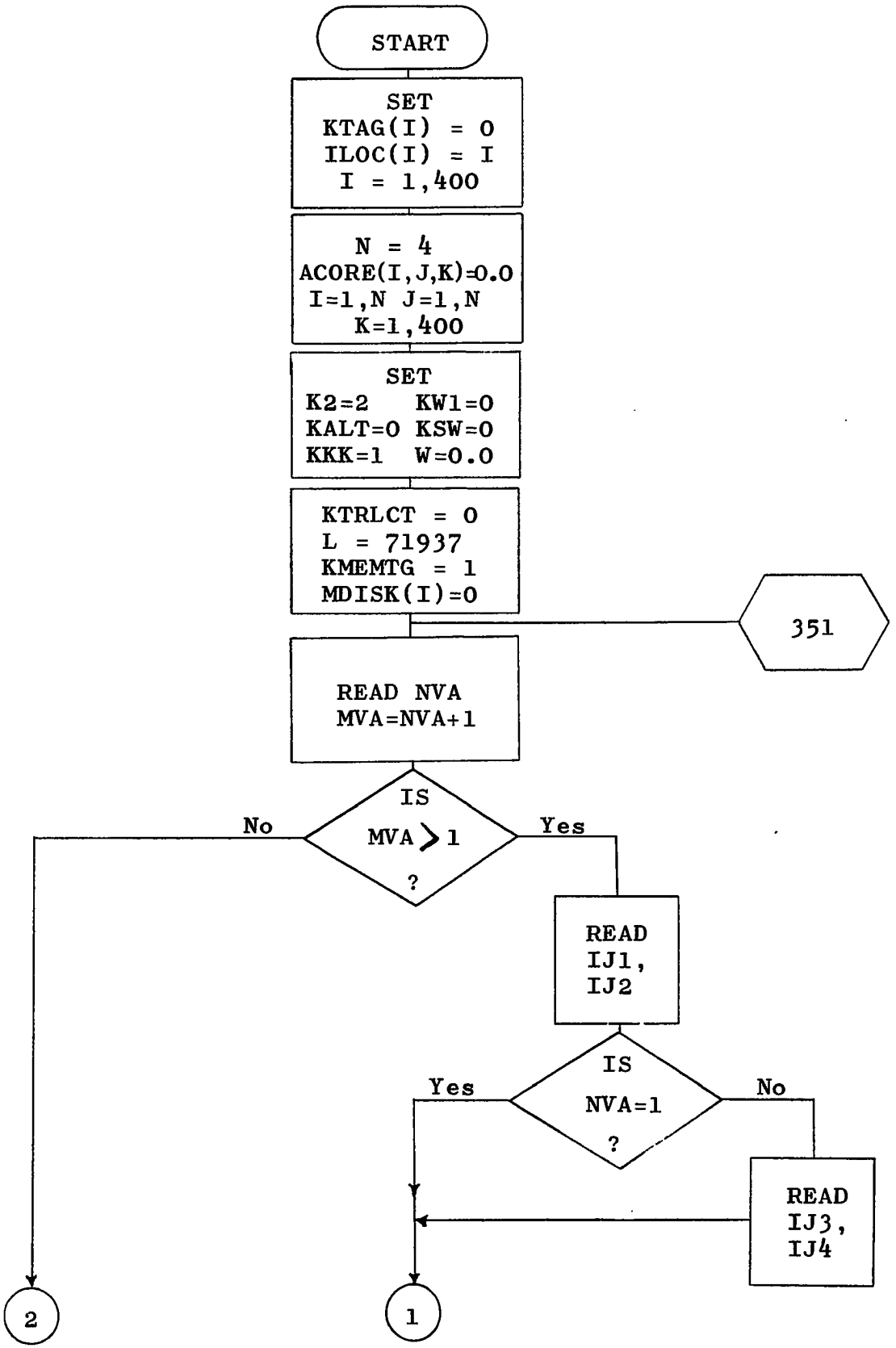
```
199 A(IJ1,IJ2)=A2(1,1)
    GO TO (205,206),NVA
206 A(IJ3,IJ4)=A2(1,2)
205 KCOUNT=KCOUNT+1
    GO TO 193
200 A(IJ1,IJ2)=A1(1,1)
    GO TO (207,208),NVA
208 A(IJ3,IJ4)=A2(1,1)
207 KCOUNT=KCOUNT+1
193 IF(KW1)799,778,799
799 KW1A=1
    KW1=0
    GO TO 99
778 IF(KW1A-2)779,780,779
779 KW1A=2
    GO TO 99
240 IF(NVR)780,780,241
241 IF(KW1)143,143,144
143 IF(IND1-KREL1)145,146,145
146 GO TO (780,231),NVR
231 IF(IND2-KREL2)145,780,145
145 KSW=1
    DO 148 I=1,N
148 Y(I)=YO(I)
    KTRLCT=KTRLCT+1
    KIND=2
    IF(KTRLCT-KTRIAL)232,233,233
232 Y(KVF1)=YCRIT(KVF1)+0.01
    GO TO 142
233 IF(KVF2)330,330,331
330 GO TO 277
331 IF(KTRLCT-KTRIAL*2)234,235,235
234 Y(KVF2)=YCRIT(KVF2)+0.01
    GO TO 142
235 GO TO (236,237),KTWOV
236 IF(KTRLCT-KTRIAL*3)238,237,237
238 Y(KVF1)=YCRIT(KVF1)+0.01
    Y(KVF2)=YCRIT(KVF2)+0.01
    GO TO 142
237 PRINT 239
239 FORMAT(1X,28H ADAPTATION HAS NOT OCCURRED)
    IND1=1
    IND2=1
    KTRLCT=0
    GO TO 103
144 KSW=1
    DO 147 I=1,N
147 Y(I)=YO(I)
    GO TO 99
780 PRINT 781
781 FORMAT(1X,24H ADAPTATION HAS OCCURRED)
```

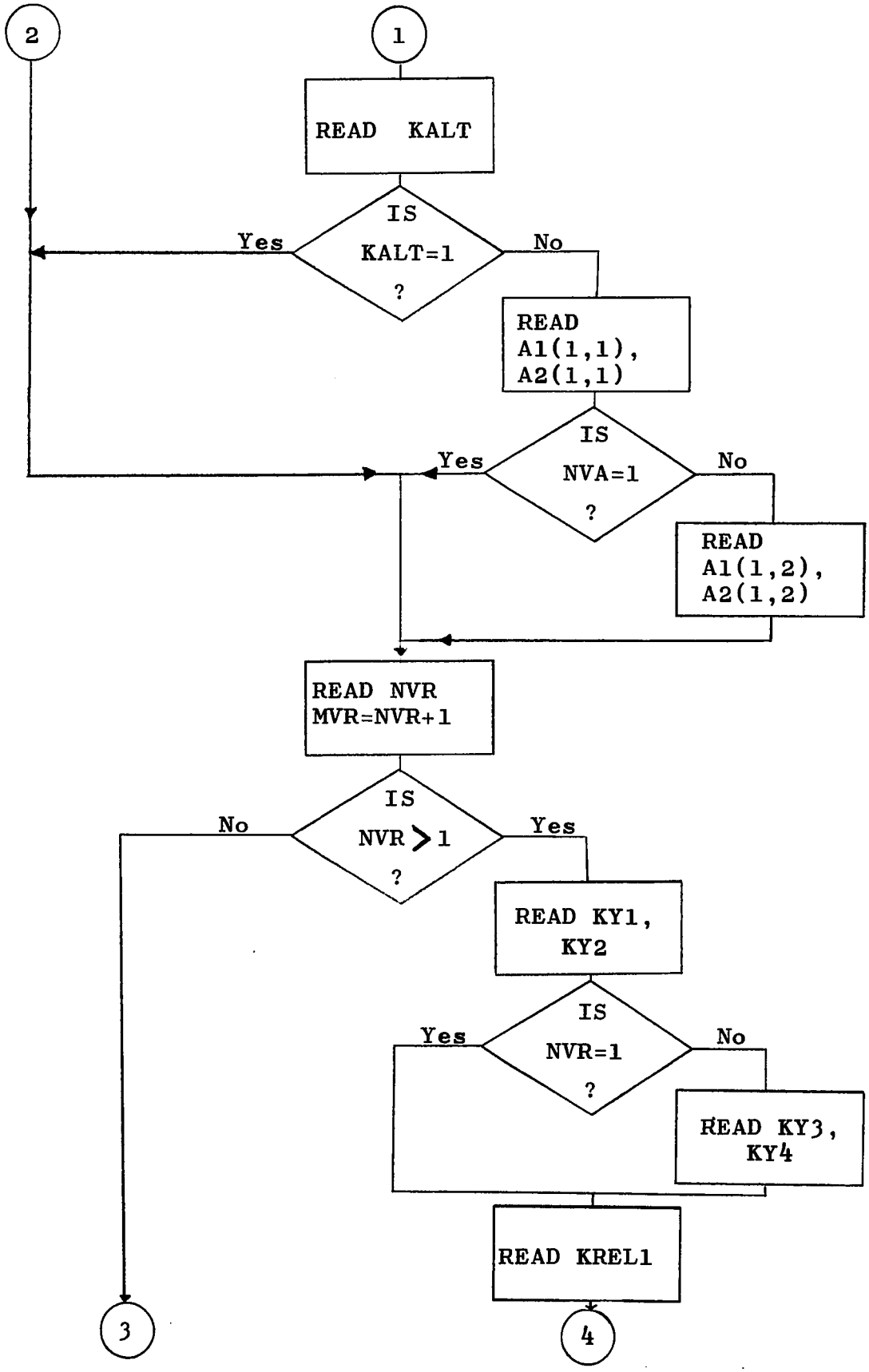
```
    PRINT 606, KW1
103 PRINT 106, KKK
106 FORMAT(1X,11H RUN NUMBER,I6,///)
    J8=1
    KW1A=1
    KKK=KKK+1
    KW1=0
    IND1=1
    IND2=1
    KSW=1
    KSWCT=0
    KTRLCT=0
    GO TO 211
111 STOP
    END
```

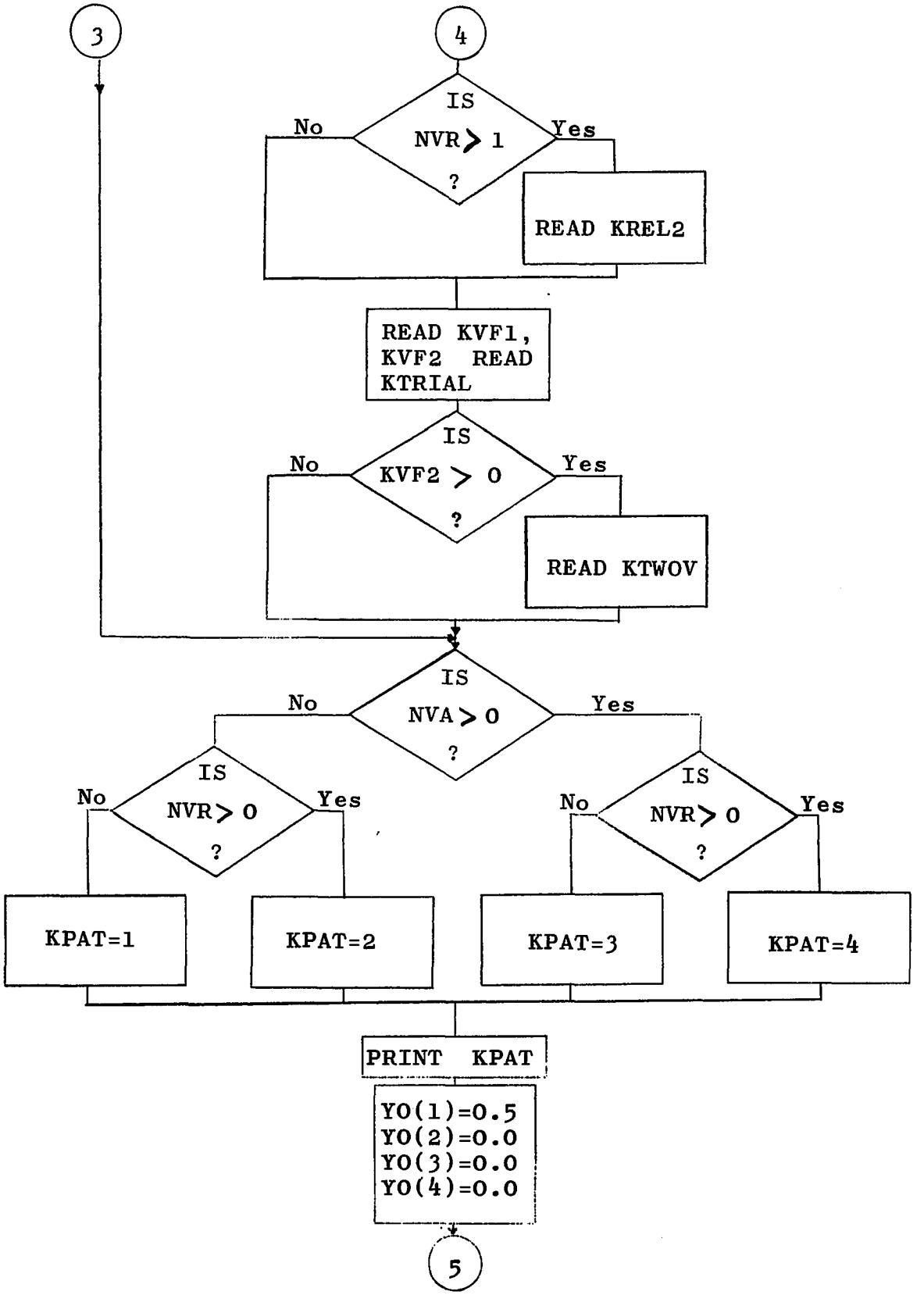
APPENDIX D

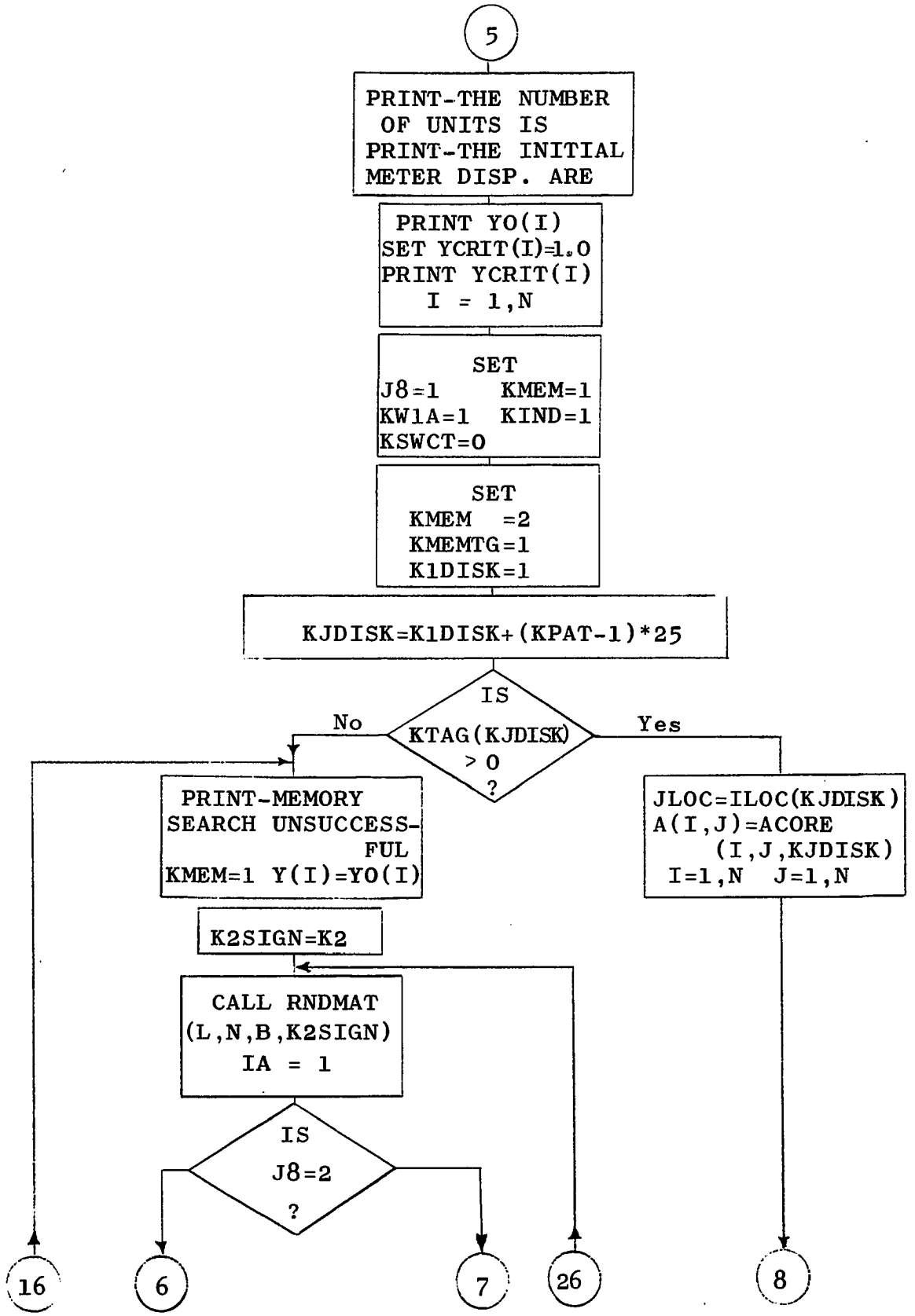
FLOW CHART AND LISTING OF THE COMPUTER PROGRAM TO SIMULATE THE HOMEOSTAT WITH MEMORY AND LEARNING

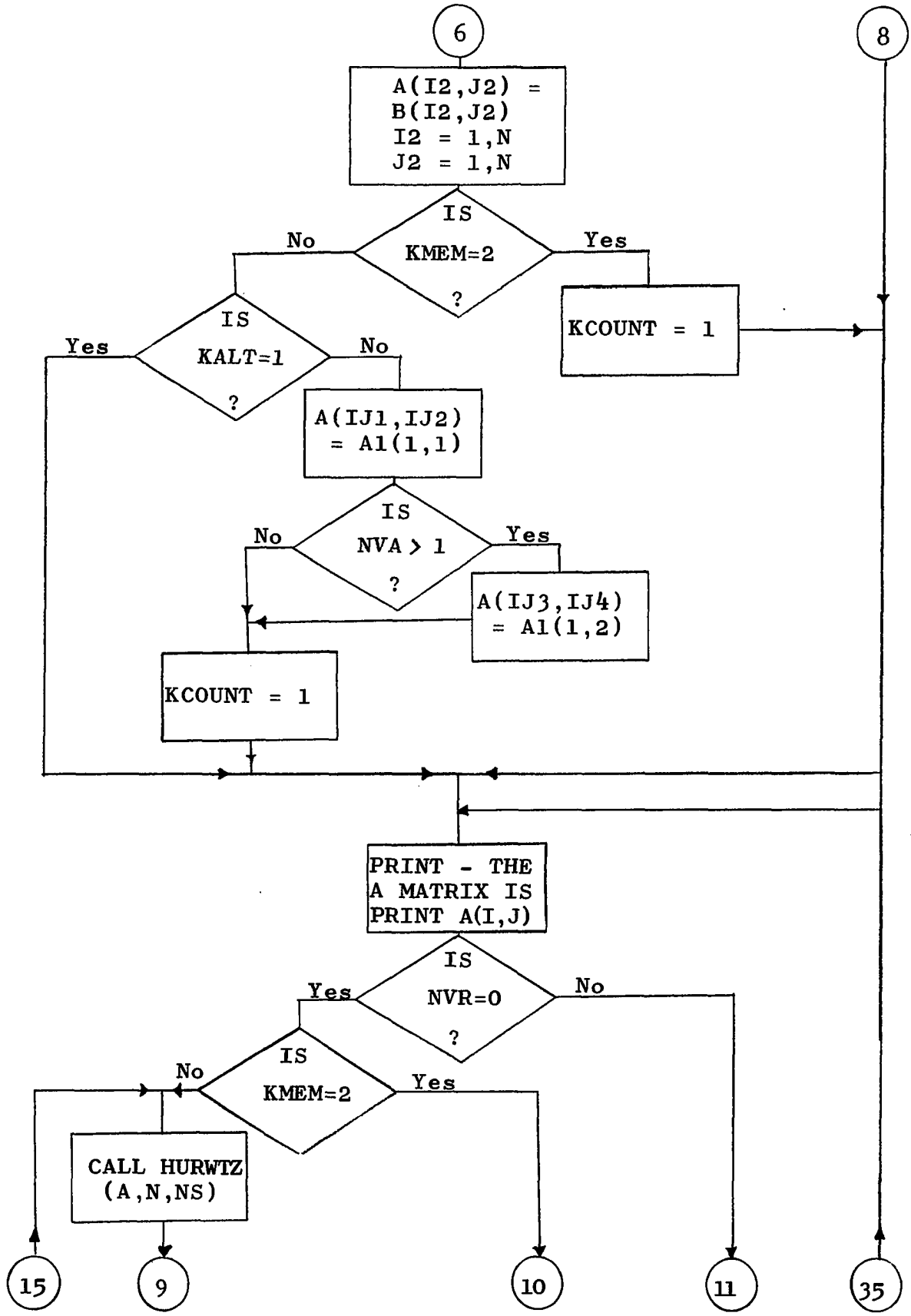
FLOW CHART FOR COMPUTER PROGRAM TO SIMULATE THE HOMEOSTAT WITH MEMORY AND LEARNING - IBM 360

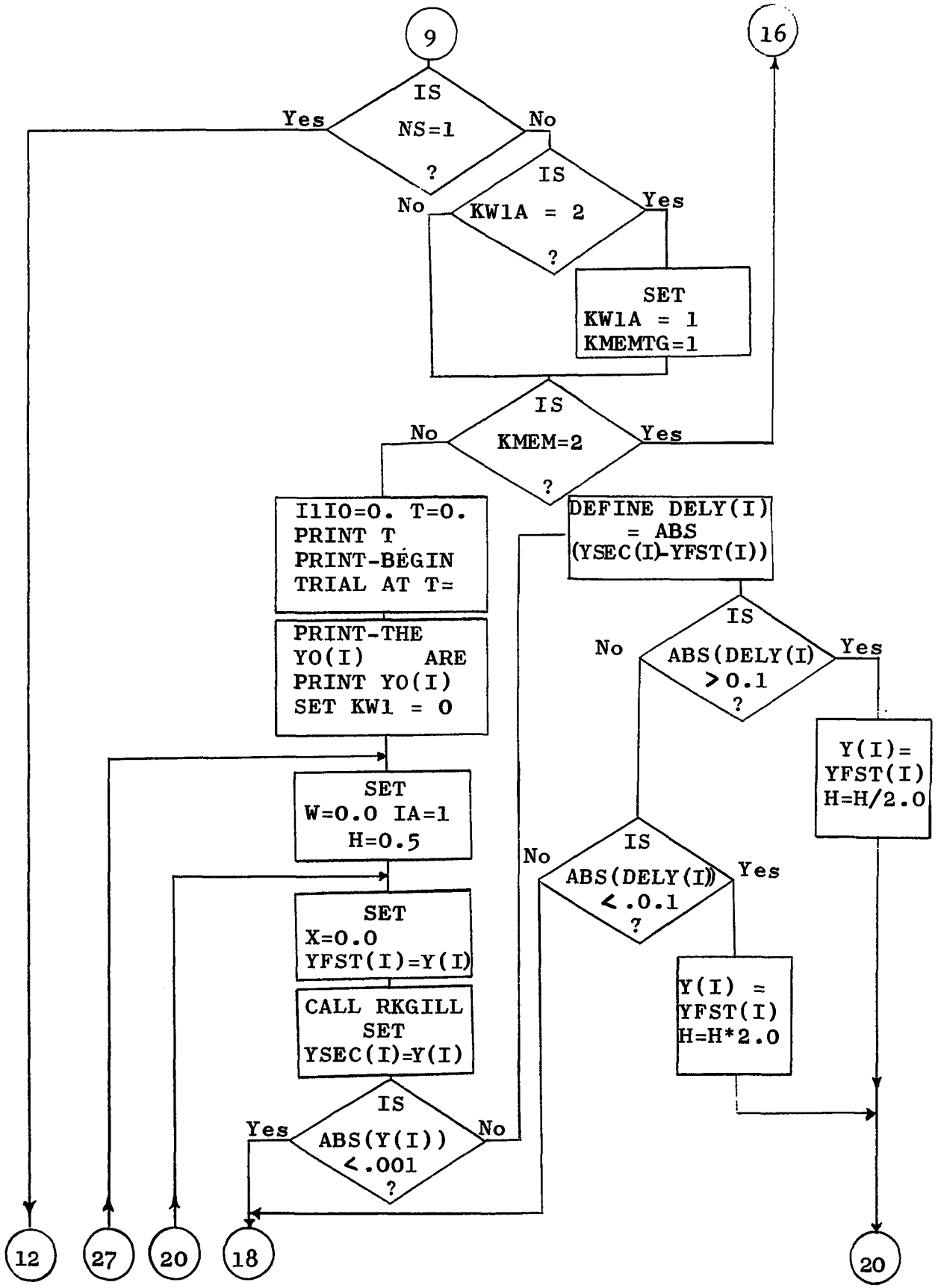


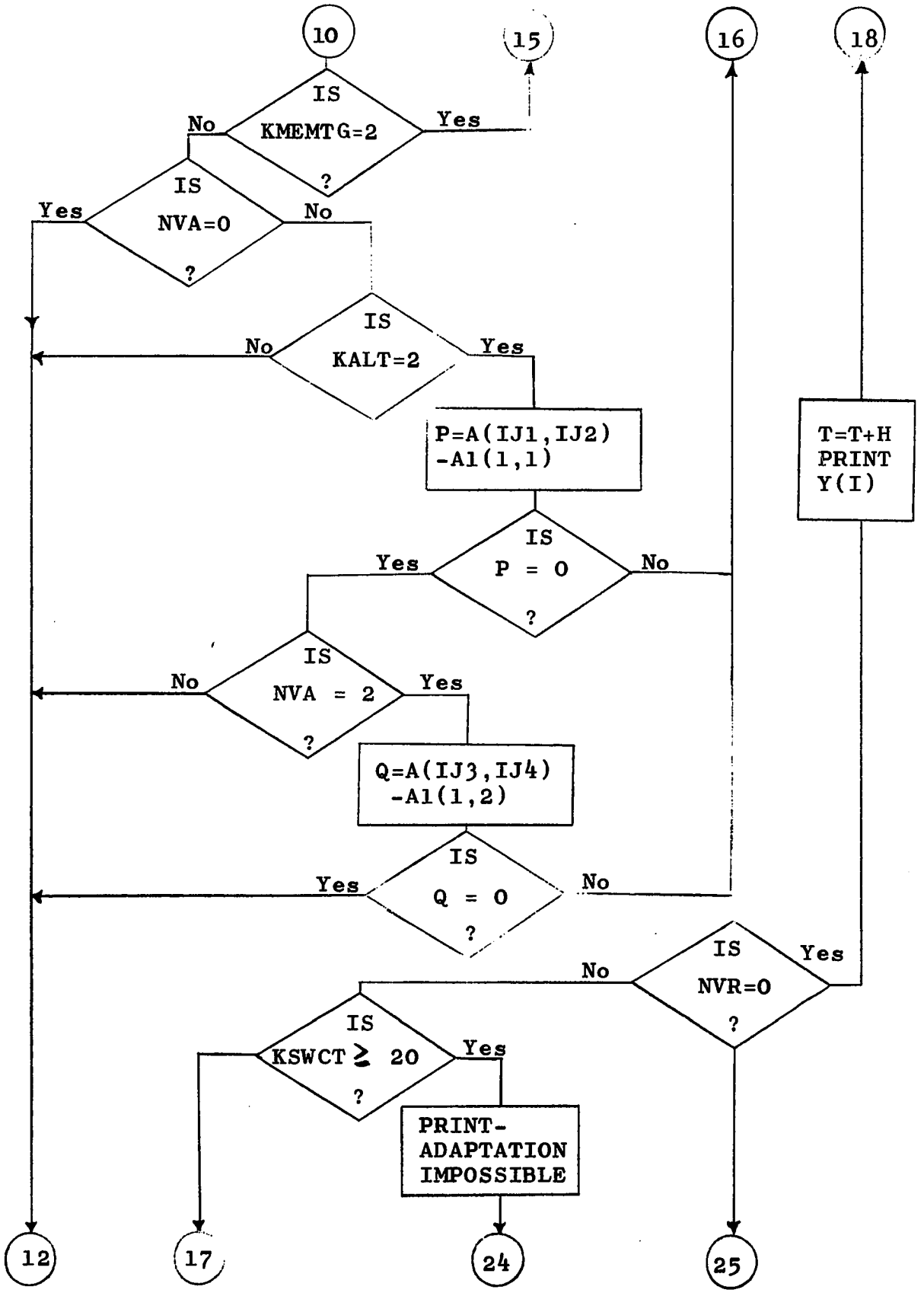

2
1
READ KALT
IS KALT=1 ?
Yes
No
READ A1(1,1), A2(1,1)
IS NVA=1 ?
Yes
No
READ A1(1,2), A2(1,2)
READ NVR
MVR=NVR+1
IS NVR > 1 ?
No
Yes
READ KY1, KY2
IS NVR=1 ?
Yes
No
READ KY3, KY4
READ KREL1
3
4

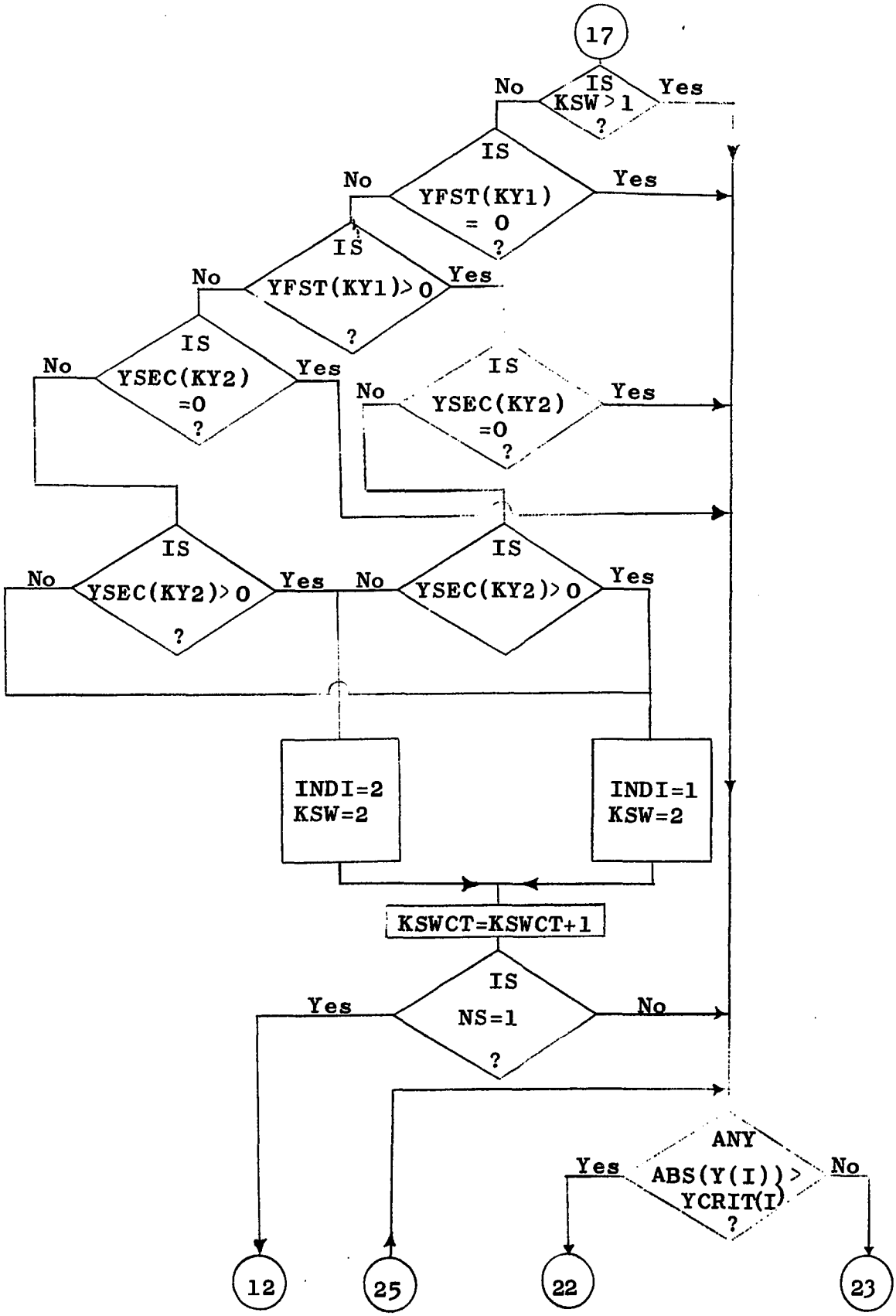

3
4
IS
NVR > 1
?
No
Yes
READ KREL2
READ KVF1,
KVF2 READ
KTRIAL
IS
KVF2 > 0
?
No
Yes
READ KTWOV
IS
NVA > 0
?
No
Yes
IS
NVR > 0
?
No
Yes
IS
NVR > 0
?
No
Yes
KPAT=1
KPAT=2
KPAT=3
KPAT=4
PRINT KPAT
YO(1)=0.5
YO(2)=0.0
YO(3)=0.0
YO(4)=0.0
5

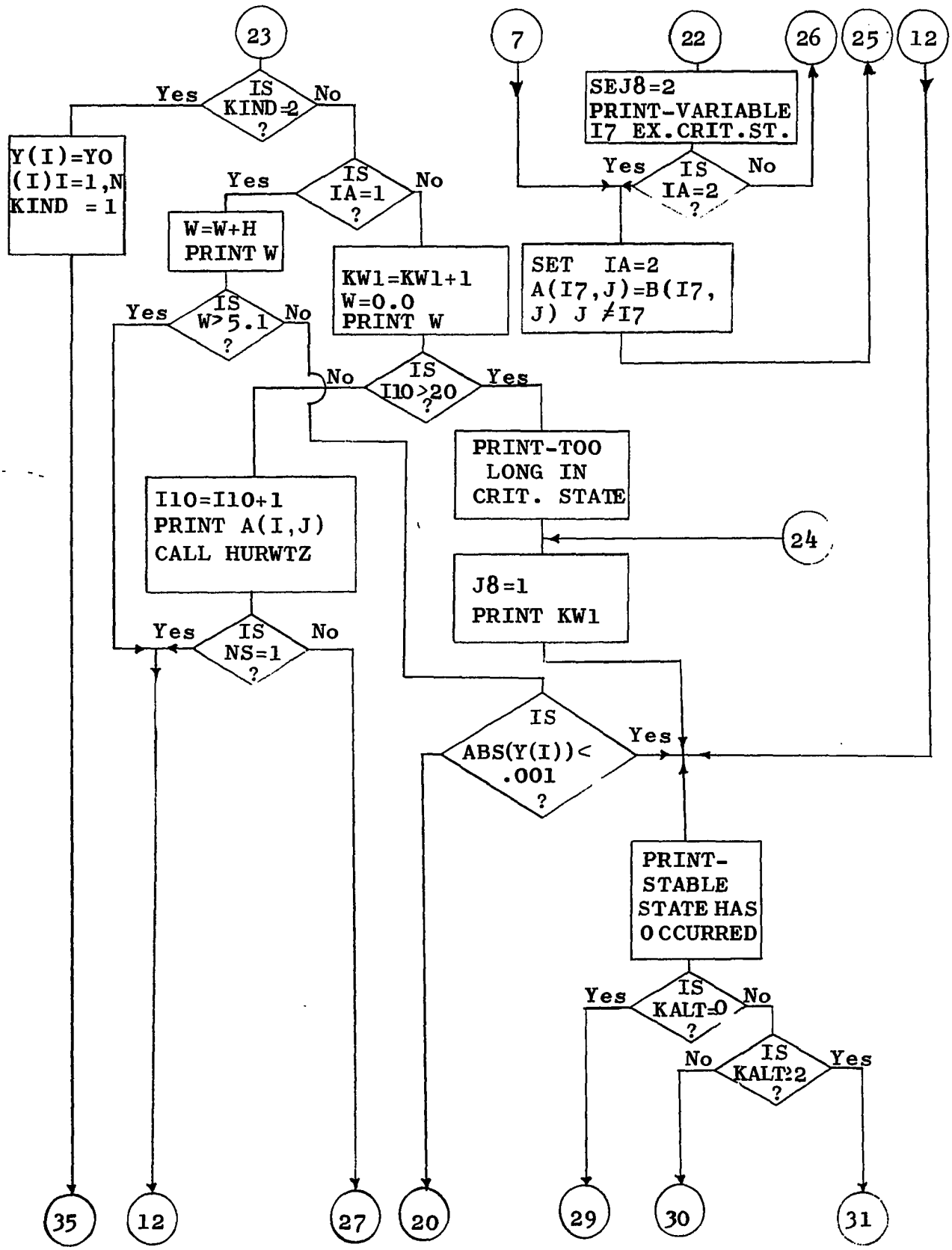

5
PRINT-THE NUMBER OF UNITS IS
PRINT-THE INITIAL METER DISP. ARE
PRINT YO(I)
SET YCRIT(I)=1.0
PRINT YCRIT(I)
I = 1,N
SET
J8=1 KMEM=1
KW1A=1 KIND=1
KSWCT=0
SET
KMEM =2
KMEMTG=1
K1DISK=1
KJDISK=K1DISK+(KPAT-1)*25
IS
KTAG(KJDISK)
> 0
?
No
Yes
PRINT-MEMORY SEARCH UNSUCCESS-FUL
KMEM=1 Y(I)=YO(I)
JLOC=ILOC(KJDISK)
A(I,J)=ACORE
(I,J,KJDISK)
I=1,N J=1,N
K2SIGN=K2
CALL RNDMAT
(L,N,B,K2SIGN)
IA = 1
IS
J8=2
?
16
6
7
26
8

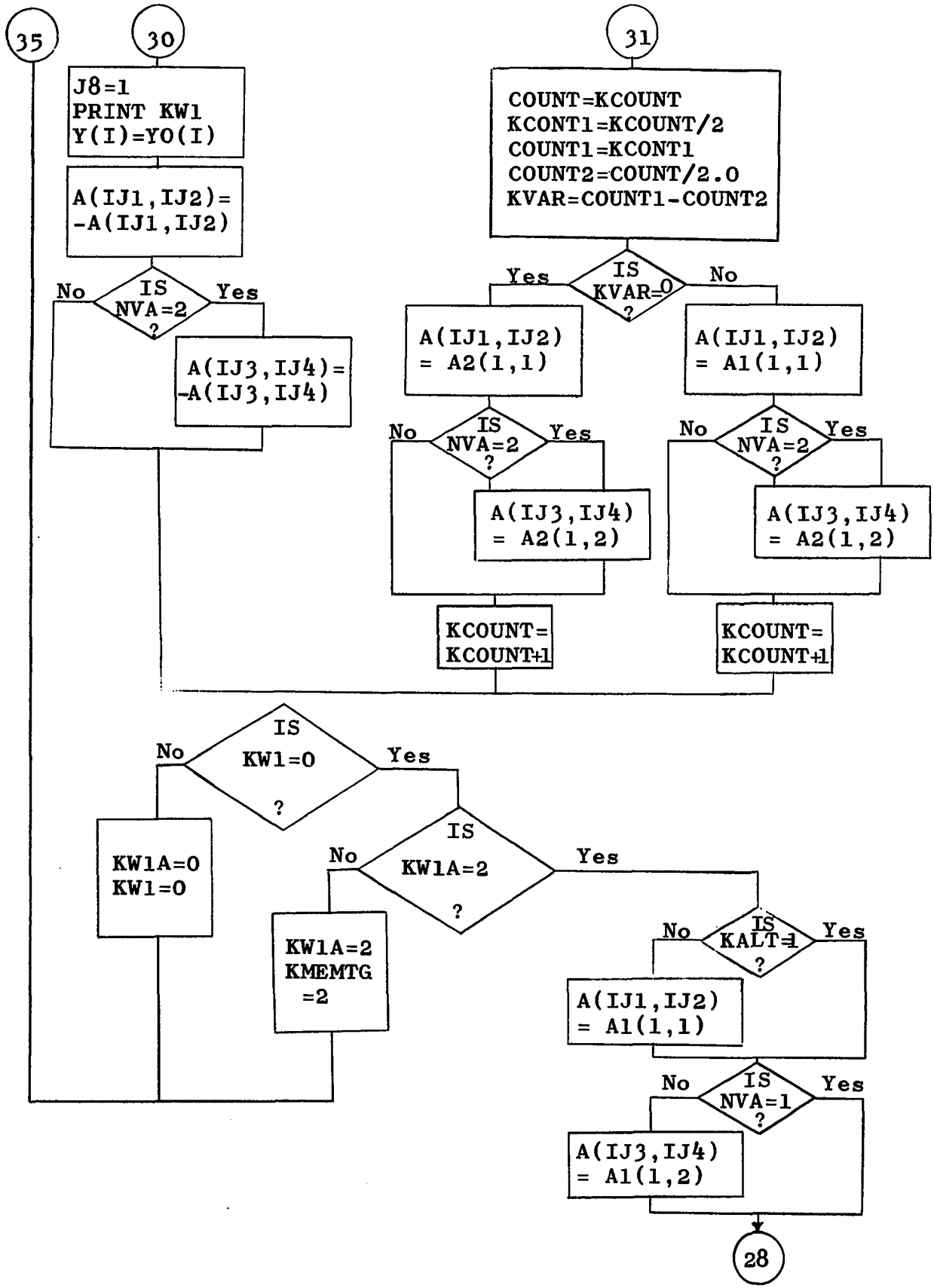

6
8
A(I2,J2) = B(I2,J2) I2 = 1,N J2 = 1,N
IS KMEM=2 ?
No
Yes
KCOUNT = 1
IS KALT=1 ?
Yes
No
A(IJ1,IJ2) = A1(1,1)
IS NVA > 1 ?
No
Yes
A(IJ3,IJ4) = A1(1,2)
KCOUNT = 1
PRINT - THE A MATRIX IS PRINT A(I,J)
IS NVR=0 ?
Yes
No
IS KMEM=2
No
Yes
CALL HURWTZ (A,N,NS)
15
9
10
11
35

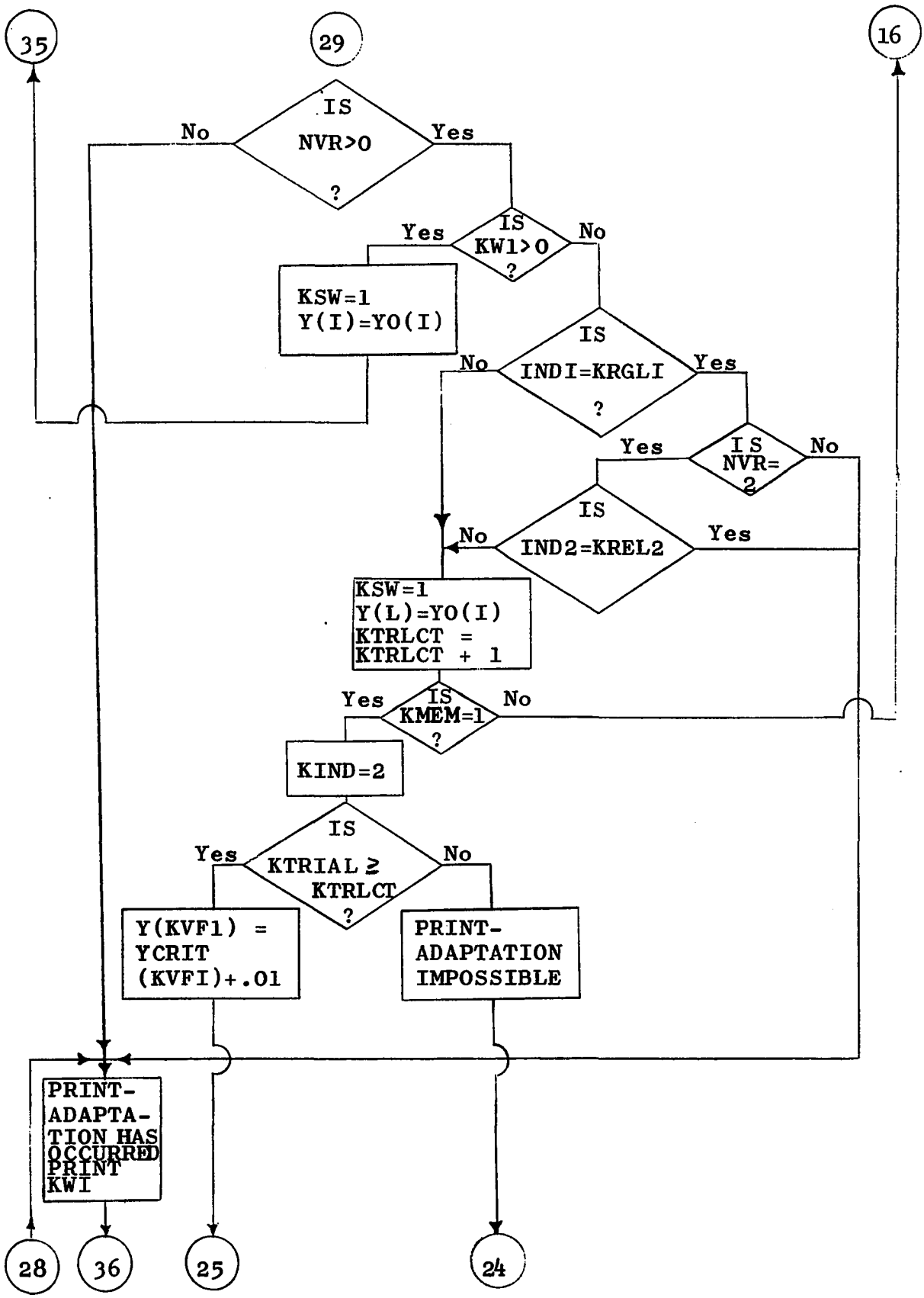

9
16
IS
NS=1
?
Yes
No
IS
KW1A = 2
?
No
Yes
SET
KW1A = 1
KMEMTG=1
IS
KMEM=2
?
No
Yes
I1IO=0. T=0.
PRINT T
PRINT-BEGIN
TRIAL AT T=
DEFINE DELY(I)
= ABS
(YSEC(I)-YFST(I))
PRINT-THE
YO(I) ARE
PRINT YO(I)
SET KW1 = 0
IS
ABS(DELY(I)
>0.1
?
No
Yes
Y(I)=
YFST(I)
H=H/2.0
SET
W=0.0 IA=1
H=0.5
IS
ABS(DELY(I))
<.0.1
?
No
Yes
SET
X=0.0
YFST(I)=Y(I)
Y(I) =
YFST(I)
H=H*2.0
CALL RKGILL
SET
YSEC(I)=Y(I)
IS
ABS(Y(I))
<.001
?
Yes
No
12
27
20
18
20

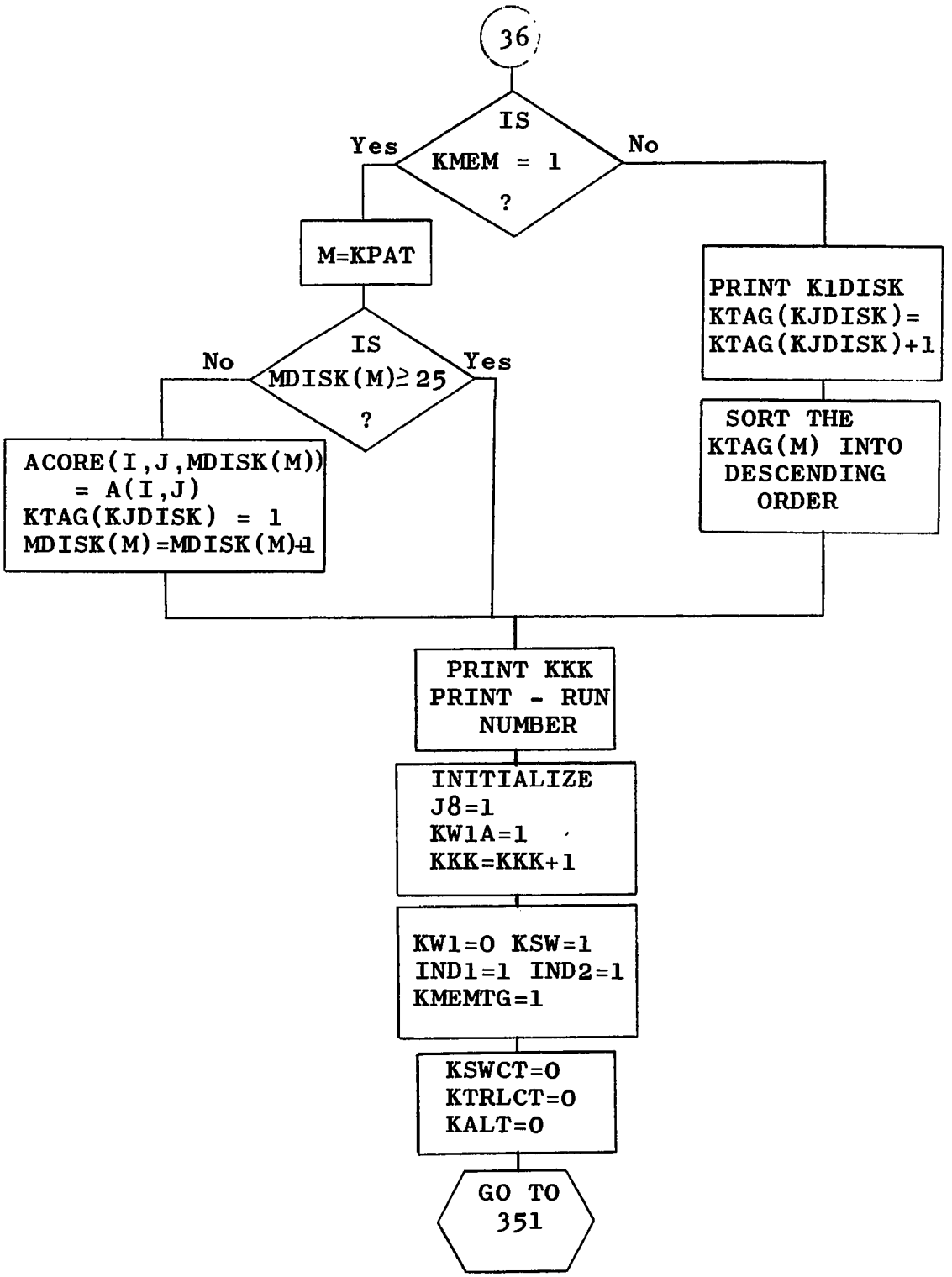

10
15
16
18
IS KMEMTG=2 ?
Yes
No
IS NVA=0 ?
Yes
No
IS KALT=2
No
Yes
P=A(IJ1,IJ2) -A1(1,1)
T=T+H PRINT Y(I)
IS P = 0 ?
Yes
No
IS NVA = 2 ?
No
Yes
Q=A(IJ3,IJ4) -A1(1,2)
IS Q = 0 ?
Yes
No
IS NVR=0 ?
Yes
No
IS KSWCT ≥ 20 ?
Yes
PRINT- ADAPTATION IMPOSSIBLE
12
17
24
25


17
IS KSW > 1 ?
No
Yes
IS YFST(KY1) = 0 ?
No
Yes
IS YFST(KY1) > 0 ?
No
Yes
IS YSEC(KY2) =0 ?
No
Yes
IS YSEC(KY2) =0 ?
No
Yes
IS YSEC(KY2) > 0 ?
No
Yes
IS YSEC(KY2) > 0 ?
No
Yes
INDI=2 KSW=2
INDI=1 KSW=2
KSWCT=KSWCT+1
IS NS=1 ?
Yes
No
ANY ABS(Y(I)) > YCRIT(I) ?
Yes
No
12
25
22
23


23
IS KIND=2 ?
Yes
No
Y(I)=YO (I)I=1,N KIND = 1
IS IA=1 ?
Yes
No
W=W+H PRINT W
IS W>5.1 ?
Yes
No
KW1=KW1+1 W=0.0 PRINT W
IS I10>20 ?
No
Yes
I10=I10+1 PRINT A(I,J) CALL HURWTZ
IS NS=1 ?
Yes
No
PRINT-TOO LONG IN CRIT. STATE
7
22
26
25
12
SEJ8=2 PRINT-VARIABLE I7 EX.CRIT.ST.
IS IA=2 ?
Yes
No
SET IA=2 A(I7,J)=B(I7, J) J ≠I7
24
J8=1 PRINT KW1
IS ABS(Y(I))< .001 ?
Yes
PRINT-STABLE STATE HAS OCCURRED
IS KALT=0 ?
Yes
No
IS KALT=2 ?
No
Yes
35
12
27
20
29
30
31


35
30
31
J8=1
PRINT KW1
Y(I)=YO(I)
A(IJ1,IJ2)=
-A(IJ1,IJ2)
No
IS
NVA=2
?
Yes
A(IJ3,IJ4)=
-A(IJ3,IJ4)
COUNT=KCOUNT
KCONT1=KCOUNT/2
COUNT1=KCONT1
COUNT2=COUNT/2.0
KVAR=COUNT1-COUNT2
Yes
IS
KVAR=0
?
No
A(IJ1,IJ2)
= A2(1,1)
A(IJ1,IJ2)
= A1(1,1)
No
IS
NVA=2
?
Yes
A(IJ3,IJ4)
= A2(1,2)
No
IS
NVA=2
?
Yes
A(IJ3,IJ4)
= A2(1,2)
KCOUNT=
KCOUNT+1
KCOUNT=
KCOUNT+1
No
IS
KW1=0
?
Yes
KW1A=0
KW1=0
No
IS
KW1A=2
?
Yes
KW1A=2
KMEMTG
=2
No
IS
KALT=1
?
Yes
A(IJ1,IJ2)
= A1(1,1)
No
IS
NVA=1
?
Yes
A(IJ3,IJ4)
= A1(1,2)
28


35
29
16
IS NVR>0 ?
No
Yes
IS KW1>0 ?
Yes
No
KSW=1
Y(I)=YO(I)
IS INDI=KRGLI ?
No
Yes
IS NVR=2
Yes
No
IS IND2=KREL2
No
Yes
KSW=1
Y(L)=YO(I)
KTRLCT = KTRLCT + 1
IS KMEM=1 ?
Yes
No
KIND=2
IS KTRIAL ≥ KTRLCT ?
Yes
No
Y(KVF1) = YCRIT (KVFI)+.01
PRINT-ADAPTATION IMPOSSIBLE
PRINT-ADAPTA-TION HAS OCCURRED PRINT KWI
28
36
25
24


36
IS KMEM = 1 ?
Yes
No
M=KPAT
PRINT K1DISK
KTAG(KJDISK)=
KTAG(KJDISK)+1
IS MDISK(M)≥25 ?
No
Yes
SORT THE KTAG(M) INTO DESCENDING ORDER
ACORE(I,J,MDISK(M)) = A(I,J)
KTAG(KJDISK) = 1
MDISK(M)=MDISK(M)+1
PRINT KKK
PRINT - RUN NUMBER
INITIALIZE
J8=1
KW1A=1
KKK=KKK+1
KW1=0 KSW=1
IND1=1 IND2=1
KMEMTG=1
KSWCT=0
KTRLCT=0
KALT=0
GO TO 351

```
C       PROGRAM TEST
        EXTERNAL FUNC
        DIMENSION A(5,5),B(5,5),C(5,5),D(5,5)
        DIMENSION YO(5),Y(5),YCRIT(5)
        DIMENSION YFST(5),YSEC(5),DELY(5),A1(1,2),A2(1,2)
        DIMENSION KTAG(400),ILOC(400)
        DIMENSION ACORE(5,5,100)
        DO 340 I=1,400
  340   ILOC(I)=I
        DO 357 I=1,400
  357   KTAG(I)=0
        DO 361 I=1,5
        DO 361 J=1,5
        DO 361 K=1,100
  361   ACORE(I,J,K)=0.0
        N=4
        KW1=0
        KMEMTG=1
        KKK=1
        W=0.0
        K2=2
    1   FORMAT(I3)
        KALT=0
        KTRLCT=0
        KSW=1
        MDISK1=0
        MDISK2=0
        MDISK3=0
        MDISK4=0
        L=71937
  351   READ(1,161)NVA
  161   FORMAT(I3)
        MVA=NVA&1
        GO TO (162,163,163),MVA
  163   READ(1,165)IJ1,IJ2
  165   FORMAT(2I3)
        GO TO (164,166),NVA
  166   READ(1,165)IJ3,IJ4
  164   READ(1,161)KALT
        GO TO (162,168),KALT
  168   READ(1,265)A1(1,1),A2(1,1)
  265   FORMAT(2F4.0)
        GO TO (162,170),NVA
  170   READ(1,265)A1(1,2),A2(1,2)
        GO TO 162
  162   READ(1,161)NVR
        MVR=NVR&1
        GO TO (171,172,172),MVR
  172   READ(1,165)KY1,KY2
```

```
      GO TO (173,174),NVR
  174 READ(1,165)KY3,KY4
  173 READ(1,161)KREL1
      GO TO (176,175),NVR
  175 READ(1,161)KREL2
03/24/67              FORTMAIN
  176 READ(1,165)KVF1,KVF2
      READ(1,165)KTRIAL
      IF(KVF2)171,171,332
  332 READ(1,161)KTWOV
      GO TO 171
  171 IF(NVA)180,180,181
  180 IF(NVR)182,182,183
  182 KPAT=1
      GO TO 190
  183 KPAT=2
      GO TO 190
  181 IF(NVR)184,184,185
  184 KPAT =3
      GO TO 190
  185 KPAT=4
      GO TO 190
  190 WRITE(3,309)KPAT
  309 FORMAT(1X,7H KPAT =,I3,//)
  110 YO(1)=0.5
      YO(2)=0.0
      YO(3)=0.0
      YO(4)=0.0
    4 FORMAT(1X,23H THE NUMBER OF UNITS IS,I3,//)
      WRITE(3,5)
    5 FORMAT(1X,36H THE INITIAL METER DISPLACEMENTS ARE, //)
      WRITE(3,6)(YO(I),1=1,N)
    6 FORMAT(1X,10F8.2,///)
      YCRIT(1)=0.9
      YCRIT(2)=0.75
      YCRIT(3)=0.75
      YCRIT(4)=1.0
      WRITE(3,201)
  201 FORMAT(///,24H THE CRITICAL STATES ARE,//)
      WRITE(3,6)(YCRIT(I),I=1,N)
      J8=1
      KMEM=1
      KW1A=1
      KSWCT=0
      KIND=1
  294 KMEM=2
      DO 295 K1DISK=1,25
      KJDISK=K1DISK&(KPAT-1)*25
      IF(KTAG(KJDISK))402,352,402
  402 JLOC=ILOC(KJDISK)
```

```
      DO 358 I=1,N
      DO 358 J=1,N
  358 A(I,J)=ACORE(I,J,JLOC)
      GO TO 99
  295 CONTINUE
  352 WRITE(3,314)
  314 FORMAT(1X,27H MEMORY SEARCH UNSUCCESSFUL,//)
      KMEM=1
  211 DO 7 I=1,N
    7 Y(I)=YO(I)
03/24/67              FORTMAIN
      K2SIGN=K2
   13 CALL RNDMAT(L,N,B,K2SIGN)
  342 IA=1
   36 GO TO(71,72),J8
   71 DO 41 I2=1,N
      DO 41 J2=1,N
   41 A(I2,J2)=B(I2,J2)
      GO TO (711,203),KMEM
  711 IF(KALT-1)99,99,202
  202 A(IJ1,IJ2)=A1(1,1)
      GO TO (203,204),NVA
  204 A(IJ3,IJ4)=A1(1,2)
  203 KCOUNT=1
   99 WRITE(3,433)
  433 FORMAT(///,16H THE A MATRIX IS,//)
      DO 131 I=1,N
      WRITE(3,132)(A(I,J),J=1,N)
  132 FORMAT(1X,10F8.2)
  131 CONTINUE
      IF(NVR)250,250,396
  250 GO TO (701,703),KMEM
  701 CALL HURWTZ(A,N,NS)
      IF(NS-1)81,81,383
  383 GO TO (310,384),KW1A
  384 KW1A=1
      KMEMTG=1
      GO TO 310
  703 GO TO (380,701),KMEMTG
  380 IF(NVA)81,81,704
  704 IF(KALT-2)81,705,705
  705 IF(A(IJ1,IJ2)-A1(1,1))708,706,708
  706 GO TO (31,707),NVA
  707 IF(A(IJ3,IJ4)-A1(1,2))708,81,708
  708 GO TO 295
  310 GO TO (396,295),KMEM
  396 I10=1
      T=0.0
      WRITE(3,450)T
  450 FORMAT(///,18H BEGIN TRIAL AT T=,F10.8)
```

```
      WRITE(3,451)
  451 FORMAT(///,14H THE YO(I) ARE)
      WRITE(3,6)(YO(I),I=1,N)
      KW1=0
   76 W=0.0
      IA=1
      H=0.05
   62 X=0.0
      DO 635 I=1,N
  635 YFST(I)=Y(I)
      CALL RKGIL1(X,H,Y,N,FUNC,A)
      DO 636 I=1,N
  636 YSEC(I)=Y(I)
      DO 480 I=1,N
      IF(ABS(Y(I))-0.001)480,480,481
  480 CONTINUE
03/24/67              FORTMAIN
      GO TO 469
  481 DO 470 I=1,N
      DELY(I)=ABS(YSEC(I))-ABS(YFST(I))
      IF(ABS(DELY(I))-0.1)470,470,463
  470 CONTINUE
      GO TO 462
  463 H=H/2.0
  467 DO 466 I=1,N
  466 Y(I)=YFST(I)
      GO TO 62
  462 DO 465 I=1,N
      IF(ABS(DELY(I))-0.01)465,469,469
  465 CONTINUE
      H=H*2.0
      GO TO 467
  469 T=T&H
      WRITE(3,452)T
  452 FORMAT(1X,6H AT T=,F20.8,14H  THE Y(I) ARE,//)
      WRITE(3,6)(Y(I),I=1,N)
      IF(NVR)142,142,212
  212 IF(KSWCT-20)276,276,277
  277 WRITE(3,288)
  288 FORMAT(1X,22H ADAPTATION IMPOSSIBLE)
      GO TO 289
  276 IF(KSW-1)141,141,142
  141 IF(YFST(KY1))121,122,123
  121 IF(YSEC(KY2))126,122,124
  126 IND1=1
      KSW=2
      GO TO (125,213),NVR
  124 IND1=2
      KSW=2
      GO TO (125,213),NVR
```

```
  123 IF(YSEC(KY2))124,122,126
  122 GO TO 142
  213 IF(YFST(KY3))214,215,216
  215 KSW=1
      GO TO 142
  214 IF(YSEC(KY4))217,215,218
  217 IND2=1
      GO TO 230
  218 IND2=2
      GO TO 230
  216 IF(YSEC(KY4))218,215,217
  230 KSW=2
  125 KSWCT=KSWCT&1
      IF(NS-1)81,81,142
  142 DO 52 I7=1,N
      IF(ABS(YCRIT(I7))-ABS(Y(I7)))54,54,52
   52 CONTINUE
      GO TO (485,275),KIND
  275 DO 348 I=1,N
  348 Y(I)=YO(I)
      KIND=1
      GO TO 99
03/24/67            FORTMAIN
  485 GO TO(631,610),IA
  631 W=W&H
      WRITE(3,499)W
  499 FORMAT(1X,3H W=,F20.8,//)
      GO TO 61
   54 WRITE(3,102)I7
  102 FORMAT(1X,3H X(,I2,33H) HAS EXCEEDED ITS CRITICAL
        STATE,//)
      J8=2
      GO TO(13,72),IA
  610 KW1=KW1&1
      W=0.0
      WRITE(3,499)W
      IF(I10-20)91,92,92
   92 WRITE(3,93)
   93 FORMAT(1X,20H TOO LONG IN CRIT ST)
  289 J8=1
      WRITE(3,606)KW1
  606 FORMAT(1X,I5,24H   STEP CHANGES OCCURRED,//)
      GO TO 103
   91 I10=I10&1
      GO TO 611
   72 DO 73 I=1,N
      IF(I7-I)74,75,74
   75 GO TO 73
   74 A(I7,I)=B(I7,I)
   73 CONTINUE
```

```
      IA=2
      GO TO 52
  611 WRITE(3,433)
      DO 151 I=1,N
      WRITE(3,132)(A(I,J),J=1,N)
  151 CONTINUE
      GO TO 395
  395 CALL HURWTZ(A,N,NS)
      IF(NS-1)81,81,76
   61 IF(W-5.1)82,82,81
   82 DO 473 I=1,N
      IF(ABS(Y(I))-0.001)473,473,472
  473 CONTINUE
      GO TO 81
  472 GO TO 62
   81 WRITE(3,83)
   83 FORMAT(1X,26H STABLE STATE HAS OCCURRED)
      IF(KALT-1)240,192,198
  192 J8=1
      WRITE(3,606)KW1
      DO 776 I=1,N
  776 Y(I)=YO(I)
      A(IJ1,IJ2)=-A(IJ1,IJ2)
      GO TO (193,194),NVA
  194 A(IJ3,IJ4)=-A(IJ3,IJ4)
      GO TO 193
  198 COUNT=KCOUNT
      KCONT1=KCOUNT/2
      COUNT1=KCONT1
03/24/67              FORTMAIN
      COUNT2=COUNT/2.0
      KVAR=COUNT1-COUNT2
      IF(KVAR)200,199,200
  199 A(IJ1,IJ2)=A2(1,1)
      GO TO (205,206),NVA
  206 A(IJ3,IJ4)=A2(1,2)
  205 KCOUNT=KCOUNT&1
      GO TO 193
  200 A(IJ1,IJ2)=A1(1,1)
      GO TO (207,208),NVA
  208 A(IJ3,IJ4)=A2(1,1)
  207 KCOUNT=KCOUNT&1
  193 IF(KW1)799,778,799
  799 KW1A=1
      KW1=0
      GO TO 99
  778 IF(KW1A-2)779,381,779
  381 GO TO (385,386),KALT
  386 A(IJ1,IJ2)=A1(1,1)
  385 GO TO (780,382),NVA
```

```
  382 A(IJ3,IJ4)=A1(1,2)
      GO TO 780
  779 KW1A=2
      KMEMTG=2
      GO TO 99
  240 IF(NVR)780,780,241
  241 IF(KW1)143,143,144
  143 IF(IND1-KREL1)145,146,145
  146 GO TO (780,231),NVR
  231 IF(IND2-KREL2)145,780,145
  145 KSW=1
      DO 148 I=1,N
  148 Y(I)=YO(I)
      KTRLCT=KTRLCT&1
      GO TO (311,295),KMEM
  311 KIND=2
      IF(KTRLCT-KTRIAL)232,233,233
  232 Y(KVF1)=YCRIT(KVF1)&0.01
      GO TO 142
  233 IF(KVF2)330,330,331
  330 GO TO 277
  331 IF(KTRLCT-KTRIAL*2)234,235,235
  234 Y(KVF2)=YCRIT(KVF2)&0.01
      GO TO 142
  235 GO TO (236,237),KTWOV
  236 IF(KTRLCT-KTRIAL*3)238,237,237
  238 Y(KVF1)=YCRIT(KVF1)&0.01
      Y(KVF2)=YCRIT(KVF2)&0.01
      GO TO 142
  237 WRITE(3,239)
  239 FORMAT(1X,28H ADAPTATION HAS NOT OCCURRED)
      IND1=1
      IND2=1
      KTRLCT=0
      GO TO 103
03/24/67              FORTMAIN
  144 KSW=1
      DO 147 I=1,N
  147 Y(I)=YO(I)
      GO TO 99
  780 WRITE(3,781)
  781 FORMAT(1X,24H ADAPTATION HAS OCCURRED)
      WRITE(3,606)KW1
      GO TO (350,312),KMEM
  350 GO TO (300,301,302,303),KPAT
  300 IF(MDISK1-25)304,305,305
  304 JDISK1=(KPAT-1)*25&MDISK1&1
      DO 362 I=1,N
      DO 362 J=1,N
  362 ACORE(I,J,JDISK1)=A(I,J)
```

```
      KTAG(JDISK1)=1
      MDISK1=MDISK1&1
      GO TO 305
  301 IF(MDISK2-25)306,305,305
  306 JDISK2=(KPAT-1)*25&MDISK2&1
      DO 363 I=1,N
      DO 363 J=1,N
  363 ACORE(I,J,JDISK2)=A(I,J)
      KTAG(JDISK2)=1
      MDISK2=MDISK2&1
      GO TO 305
  302 IF(MDISK3-25)307,305,305
  307 JDISK3=(KPAT-1)*25&MDISK3&1
      DO 364 I=1,N
      DO 364 J=1,N
  364 ACORE(I,J,JDISK3)=A(I,J)
      KTAG(JDISK3)=1
      MDISK3=MDISK3&1
      GO TO 305
  303 IF(MDISK4-25)308,305,305
  308 JDISK4=(KPAT-1)*25&MDISK4&1
      DO 365 I=1,N
      DO 365 J=1,N
  365 ACORE(I,J,JDISK4)=A(I,J)
      KTAG(JDISK4)=1
      MDISK4=MDISK4&1
  305 CONTINUE
      GO TO 103
  312 WRITE(3,313)K1DISK
  313 FORMAT(1X,28H ADAPTATION ON MEMORY SEARCH,I6,//)
      KTAG(KJDISK)=KTAG(KJDISK)&1
      DO 403 IKJDSK=1,25
      IKKDSK=KJDISK-IKJDSK
      IF(IKKDSK-(KPAT-1)*25)103,103,343
  343 IF(KTAG(KJDISK)-KTAG(IKKDSK))403,403,404
  404 KTEMP1=KTAG(KJDISK)
      KTEMP2=ILOC(KJDISK)
      KTAG(KJDISK)=KTAG(IKKDSK)
      ILOC(KJDISK)=ILOC(IKKDSK)
      KTAG(IKKDSK)=KTEMP1
      ILOC(IKKSDK)=KTEMP2
03/24/67              FORTMAIN
  403 CONTINUE
  103 WRITE(3,106)KKK
  106 FORMAT(1X,11H RUN NUMBER,I6,///)
      J8=1
      KW1A=1
      KKK=KKK&1
      KMEMTG=1
      KALT=0
```

```
      KW1=0
      IND1=1
      IND2=1
      KSW=1
      KSWCT=0
      KTRLCT=0
      GO TO 351
  111 STOP
      END
```