CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

GENERALIZED INTERACTIVE NETWORK SIMULATOR
(GINS)

A thesis submitted in partial satisfaction of the requirements for the degree of Masters of Science in Engineering

by

Robin Roland Johnson

August 1976
A thesis of Robin Roland Johnson is approved:

California State University, Northridge

July 1, 1976
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td></td>
<td>viii</td>
</tr>
<tr>
<td>Chapter 1:</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2:</td>
<td>Networks &amp; Simulation</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 3:</td>
<td>Generalized Interactive Network Simulator (GINS)</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 4:</td>
<td>Validation &amp; Applications</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 5:</td>
<td>Conclusion</td>
<td>42</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A:</td>
<td>System Image &amp; Data Manipulation</td>
<td>45</td>
</tr>
<tr>
<td>B:</td>
<td>Operating Manual</td>
<td></td>
</tr>
<tr>
<td>Section 1:</td>
<td>Path Equations &amp; Tabular Data</td>
<td>56</td>
</tr>
<tr>
<td>Section 2:</td>
<td>Operating Modes</td>
<td>59</td>
</tr>
<tr>
<td>Section 3:</td>
<td>Input Subroutine</td>
<td>62</td>
</tr>
<tr>
<td>Section 4:</td>
<td>Edit Subroutine</td>
<td>69</td>
</tr>
<tr>
<td>Section 5:</td>
<td>Tabular Input Subroutine</td>
<td>75</td>
</tr>
<tr>
<td>Section 6:</td>
<td>Tabular Report Subroutine</td>
<td>77</td>
</tr>
<tr>
<td>Section 7:</td>
<td>Plot Subroutines</td>
<td>80</td>
</tr>
<tr>
<td>Section 8:</td>
<td>Error Messages</td>
<td>87</td>
</tr>
<tr>
<td>C:</td>
<td>Flow Charts</td>
<td></td>
</tr>
<tr>
<td>Top Level</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>File Level</td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>Input Subroutine 1</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>Input Subroutine 2</td>
<td></td>
<td>93</td>
</tr>
<tr>
<td>Edit Change Subroutine</td>
<td></td>
<td>94</td>
</tr>
<tr>
<td>Edit Add Subroutine 1</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>Edit Add Subroutine 2</td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>Tabular Input Subroutine</td>
<td></td>
<td>98</td>
</tr>
<tr>
<td>Processor Subroutine</td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>Tabular Report Subroutine</td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>Axis Subroutine</td>
<td></td>
<td>103</td>
</tr>
<tr>
<td>Subroutine</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Data Plot Subroutine</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Label Subroutine</td>
<td>106</td>
<td></td>
</tr>
</tbody>
</table>

D: Listings

<table>
<thead>
<tr>
<th>Listing</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Program</td>
<td>108</td>
</tr>
<tr>
<td>Input Subroutine 1</td>
<td>109</td>
</tr>
<tr>
<td>Input Subroutine 2</td>
<td>111</td>
</tr>
<tr>
<td>Edit Change Subroutine</td>
<td>113</td>
</tr>
<tr>
<td>Edit Add Subroutine 1</td>
<td>115</td>
</tr>
<tr>
<td>Edit Add Subroutine 2</td>
<td>117</td>
</tr>
<tr>
<td>Tabular Input Subroutine</td>
<td>119</td>
</tr>
<tr>
<td>Processor Subroutine</td>
<td>120</td>
</tr>
<tr>
<td>Tabular Report Subroutine</td>
<td>122</td>
</tr>
<tr>
<td>Axis Subroutine</td>
<td>126</td>
</tr>
<tr>
<td>Data Plot Subroutine</td>
<td>127</td>
</tr>
<tr>
<td>Label Subroutine</td>
<td>129</td>
</tr>
</tbody>
</table>
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1:</td>
<td>In service weapon system management effectiveness model</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2:</td>
<td>A directed network graph</td>
<td>5</td>
</tr>
<tr>
<td>Figure 3:</td>
<td>An undirected network graph</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4:</td>
<td>Incidence matrix for a directed graph</td>
<td>7</td>
</tr>
<tr>
<td>Figure 5:</td>
<td>Incidence matrix for graph in figure 2</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6:</td>
<td>Incidence matrix for graph in figure 3</td>
<td>8</td>
</tr>
<tr>
<td>Figure 7:</td>
<td>Example of a critical path</td>
<td>10</td>
</tr>
<tr>
<td>Figure 8:</td>
<td>Example network</td>
<td>15</td>
</tr>
<tr>
<td>Figure 9:</td>
<td>Matrix R</td>
<td>16</td>
</tr>
<tr>
<td>Figure 10:</td>
<td>ALWSS</td>
<td>26</td>
</tr>
<tr>
<td>Figure 11:</td>
<td>GINS representation of ALWSS</td>
<td>27</td>
</tr>
<tr>
<td>Figure 12:</td>
<td>Network states</td>
<td>28</td>
</tr>
<tr>
<td>Figure 13:</td>
<td>Network flows</td>
<td>29</td>
</tr>
<tr>
<td>Figure 14:</td>
<td>Flow equations</td>
<td>30</td>
</tr>
<tr>
<td>Figure 15:</td>
<td>Comparison of results of GINS with ALWSS</td>
<td>31</td>
</tr>
<tr>
<td>Figure 16:</td>
<td>ALWSS plot</td>
<td>32</td>
</tr>
<tr>
<td>Figure 17:</td>
<td>GINS plot</td>
<td>33</td>
</tr>
<tr>
<td>Figure 18:</td>
<td>Full adder</td>
<td>34</td>
</tr>
<tr>
<td>Figure 19:</td>
<td>GINS network for figure 18</td>
<td>35</td>
</tr>
<tr>
<td>Figure 20:</td>
<td>Flow equations</td>
<td>36</td>
</tr>
<tr>
<td>Figure 21:</td>
<td>Truth table for full adder</td>
<td>36</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>22</td>
<td>P, S, T and U arrays</td>
<td>48</td>
</tr>
<tr>
<td>23</td>
<td>The C array</td>
<td>48</td>
</tr>
<tr>
<td>24</td>
<td>Constraint mode code</td>
<td>49</td>
</tr>
<tr>
<td>25</td>
<td>Contents of Z vector</td>
<td>49</td>
</tr>
<tr>
<td>26</td>
<td>Transfer of completed array to tape file</td>
<td>51</td>
</tr>
<tr>
<td>27</td>
<td>Transfer of predefined array from tape file to core</td>
<td>51</td>
</tr>
<tr>
<td>28</td>
<td>Initialization of link array</td>
<td>53</td>
</tr>
<tr>
<td>29</td>
<td>Initialization of node array</td>
<td>53</td>
</tr>
<tr>
<td>30</td>
<td>Program Tape structure</td>
<td>54</td>
</tr>
<tr>
<td>31</td>
<td>First Data Tape structure</td>
<td>55</td>
</tr>
<tr>
<td>32</td>
<td>A network flow graph</td>
<td>56</td>
</tr>
<tr>
<td>33</td>
<td>Example flow equation</td>
<td>56</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Results from figure 8</td>
<td>20</td>
</tr>
<tr>
<td>Table 2</td>
<td>Full adder simulation</td>
<td>37</td>
</tr>
<tr>
<td>Table 3</td>
<td>Full adder simulation</td>
<td>39</td>
</tr>
</tbody>
</table>
ABSTRACT

GENERALIZED INTERACTIVE NETWORK SIMULATOR
(GINS)

by

Robin Roland Johnson
Masters of Science in Engineering

A network-flow model is described. The model is a general, dynamic, deterministic, discrete model of a network-flow graph. The model is implemented on a programmable calculator which uses BASIC. The program is interactive with the operator to provide greater flexibility and allows nonprogrammers to work the model. The program accepts any network graph of up to 100 nodes and links, along with flow equations and constraints. Output is available as printed tables or graphs.
CHAPTER 1: INTRODUCTION

The purpose of this report is to describe a new mathematical model for simulating network-flow systems, called the Generalized Interactive Network Simulator (GINS). This model was developed by the author while working in the Systems Analysis Branch of the Fleet Weapons Engineering Directorate (FWED) at the Pacific Missile Test Center, Point Mugu, California. One of the prime missions of the Systems Analysis Branch (SAB) of FWED is to:

"...provide analytical engineering support to Fleet Weapons Engineering Directorate program managers and functional supervisors for day-to-day decision making by supplying technical and management tools for this purpose ..." (1)

With this mission in mind, John A. Connor head of SAB and Clayton Valley, member of SAB, determined the need for a generalized model for simulating the various systems under FWED cognizance. These systems include all in-service non-nuclear air-launched weapon systems used by the U. S. Navy, from bombs to air-to-air guided missiles. A variety of models are currently required to fully represent such systems, as is shown in figure 1. (2)
IN-SERVICE WEAPON SYSTEM(S)
MANAGEMENT EFFECTIVENESS MODEL

WEAPON STATUS VISIBILITY
MODEL(S)
(for Management Effectiveness Applications)

INVENTORY, QUEUING & TRANSPORTATION
MODEL(S)

LEVEL OF REPAIR & PIPELINE
MODEL(S)

TACTICAL, RELIABILITY, AND GAMING
MODEL(S)

LIFE CYCLE COST AND BUDGETARY MODEL(S)

Figure: 1
In addition to determining the desirability of a general simulator, it was also determined that such a simulator should be implemented on the Hewlett Packard 9830A programmable calculator. The reasons for this decision were three-fold. First, the language used by this machine is BASIC, an interpreter language with the important capability of being interactive with the machine operator through the keyboard. The interactive capability allows the operator to communicate with the machine in English and therefore, a person with no programming background can operate the machine after a minimal amount of instruction. A second reason for implementing GINS on the HP 9830A was the high availability of this machine. Most of the time this machine is idle and available for use. Adding to the availability of the HP 9830A is its high reliability. Since the machine is small in size it can be used in almost any office, and in addition, it does not require any special environment as do medium and large scale machines. The final reason for selecting the HP 9830A for the GINS implementation was its low cost. Since it was already acquired, the only costs associated with it are those of operation. The other options available were to use one of the commercial time sharing systems or the batch processor on base. Neither of these could offer both the advantages of high availability and low turn around time of the HP 9830A.
As a result of implementing GINS on the HP 9830A, it became possible to design GINS so that a manager could, after some initial training, sit down at the machine and run simulations of his system without the help of others. By so doing, the manager can acquire a fuller understanding of simulation methods and the behavior of his particular system as various parameters and constraints are altered. It is one of the anticipated advantages of GINS that it will instruct as well as inform the user.

While the selection of the HP 9830A provided many advantages for the use of GINS, it also imposed several constraints. Since the machine is small, it has limited ability to handle large amounts of data. As a result, only small or moderate size models are allowed. Because of its limited memory, external memory devices were required which slow down the execution time, which itself is slow by electronic computer standards. Nevertheless, since one of its applications was envisioned as an interactive tutorial device for managers, the execution speed and problem size limitations were felt to be well within reason.
CHAPTER 2: NETWORKS & SIMULATION

The purpose of this chapter is to introduce the topic of networks and simulation and describe some of the previous methods of system simulation used both inside and outside FWED. Since most of the systems that SAB is concerned with include logistics and transportation problems, the first model considered was a network model. A network is a set of nodes (vertices or points) connected by links (or arcs), with each link connecting at most only two nodes. In real systems, nodes represent distinct events, locations, or states and links are the interconnections through which entities flow from one node to another. Entities are the objects of interest in the model and may be time, devices (hardware), people or money.

A network is represented by a graph, as shown in figure 2. The graph shown in figure 2 is called a directed graph.

![Figure 2: A directed network graph](image)
A directed graph is distinguished by the property of unidirectional flow, which is to say that entities may flow in only one direction, that indicated by the arrowheads on the links. Another kind of graph is an undirected graph. An undirected graph has the property of bidirectional flow, which means entities may flow in both directions along a link. Figure 3 shows the network of figure 2 as an undirected graph.

![Figure 3: An undirected graph](image)

Note that flow between nodes 2 and 3 is now represented by a single link. Obviously, an undirected graph requires fewer links to represent a system than does a directed graph.

A directed graph may be represented mathematically by an incidence matrix, in which the i nodes are represented by i rows and the j links are represented by j columns as shown in figure 4:
A coefficient of one for an entry indicates a link directed away from that node and a negative one indicates a link terminating at that node. For the network in figure 2, the incidence matrix is given in figure 5.

Note that only two entries per column exist, and all other row positions in the column are blank. In addition, the sum of each column is 0, the number of positive 1's per row indicates the number of links terminating at that node and the number of negative 1's is the number of links originating at that node. Obviously, this table fully
describes the structure of the network of figure 2.

An undirected graph may be represented by a set of nodes and a set of pairs of nodes that represent links. (6) A form of incidence matrix can also be used on an undirected graph as shown in Figure 6, which depicts the graph in Figure 3. Note that all entries are unsigned,

\[
\begin{array}{ccccc}
  & 1 & 2 & 3 & 4 & 5 \\
1 & 1 & 1 &  &  &  \\
2 & 1 & 1 & 1 & 1 &  \\
3 & 1 & 1 & 1 &  &  \\
4 & 1 & 1 &  &  &  \\
5 & 1 & 1 &  &  &  \\
\end{array}
\]

Figure 6: Incidence matrix for graph in Figure 3

since the concepts of originating and terminating nodes do not apply to the undirected graph. Also, self-loops can be specified if a 1 is entered in any of the terms comprising the main diagonal of the incidence matrix. It is also of interest to note that the number of array entries used by an incidence matrix for an undirected graph is less than for a directed graph. In the previous examples, the directed graph requires a 5 by 8 array of 40 entries, while the undirected graph requires a 5 by 5 array of 25 entries.

Other terms used in connection with networks include
paths, loops, connection and trees. A path in network parlance is a sequence of links (irrespective of direction) connecting two nodes. When the initial and terminal nodes of a path are the same, the path is called a loop. The term connection refers to a network which has paths between all pairs of nodes. A tree is a connected network with \( n \) nodes and \( n-1 \) links which has no loops.\(^4\)

Many models use a network approach because of the wide applicability of network concepts to real-world systems. These models generally fall into one of two categories, optimization models or simulation models. The similarities between these two categories are by far more numerous than the differences, but they differ in one important aspect. An optimization model associates each link or node with some value which is an index of the performance of the system, such as cost, time, or distance. The model then seeks to either minimize or maximize the sum of this value for a path or set of paths. A simulation, on the other hand, seeks to emulate the real-world system as closely as the modeler desires, but does not optimize the system's performance. A simulation usually relies on a human to optimize system performance by adjusting the model parameters on a trial-and-error basis. A simulation is typically used when the optimization method is too complex, nonexistent, or when the trial-and-error approach has an educational
Examples of network optimization models include linear programming, critical path method (CPM) and PERT. Linear programming was developed in the early 1950's and forms the basis of many network optimization models. The critical path method (CPM) is a planning network model developed by James Kelly and Morgan Walker while working for the DuPont Company in the late 1950's. If a project is described by milestones (nodes) and activities with a given duration (links), then the sequence of milestones which takes the most time is the critical path. Figure 7 illustrates the problem and the critical path. The critical path from S to F is through C, since it takes

![Diagram](image)

Figure 7: Example of a critical path

the most time and F cannot be completed until A, C, and B have been completed. Events not on the critical path have a range of time for completion. For example, event A could begin as early as the 4th week or as late as the 10th week. Each activity (or event) in a CPM model is
given a normal duration and cost, and a crash duration and cost (assuming duration can be reduced at extra cost). A least cost schedule is then produced for several project durations and the project duration yielding the least cost is selected. The CPM model is activity-oriented since it expresses results as operations to be performed. (4)

Given the same network as described above, another model called an event-oriented model can be developed. PERT is such a model. PERT was developed at the same time as CPM in the Special Projects Office of the U.S. Navy and was first used in the Polaris program in 1958. In PERT, each activity (link) is expressed by three time estimates, optimistic, pessimistic, and likely. This time spread is usually expressed as a distribution with a mean and a variance, and determines the probability of reaching milestones on schedule. The results are expressed as a list of milestones and the probabilities of attaining them. Unlike CPM, PERT does not consider costs in determining the desired schedule. (4)

Models can be categorized by their function, structure, how they handle time and uncertainty, and their degree of generality. Functionally, models can be descriptive, predictive, or normative. A descriptive model simply provides a description of a system without predicting its behaviour or judging its performance. A predictive model relates the dependent and independent
variables so that "what if" questions may be answered. A normative model suggests a course of action such as optimization of system performance. Structurally, models can be iconic, analog, or symbolic. An iconic model retains some of the physical characteristics of the system while an analog model substitutes components or processes to provide a parallel with the system. An example of an analog model would be an electrical representation of a mechanical system. A symbolic model uses symbols to replace reality, and is typified by digital computer programs used to simulate systems.

Another major distinction between models is how they treat time and uncertainty. A model can be static, in which case it ignores system changes due to time, or it can be dynamic, in which case time is an independent variable in the model. A deterministic model essentially ignores uncertainty by yielding a unique output for a given input, while a probabilistic model yields a range of outputs for a given input. Finally, a model is categorized according to its scope. A general model is applicable to several functional areas while a specialized model is applicable only to a specific problem. (5)

The actual method of developing a system model can be described by nine basic steps. The first step is to identify and formulate the decision in writing. The second step is to identify and verbally define
constants, parameters and variables used to describe the system, then represent these with symbols. The third step is to select the most influential variables for inclusion in the model so as to keep the model simple, and distinguish between the controlable and uncontrolable variables. The fourth step is to state relationships among the variables and make assumptions or predictions on the behavior of the uncontrolable variables. The fifth step is to combine relationships into symbolic relationships, thereby forming the model. The sixth step is to manipulate the symbols. The seventh step is to derive solutions, and the eighth step is to test the model by making predictions and checking them against real-world data. The final step is to make any necessary revisions to the model.(5)

Most of the previous models used by SAB were of a specific nature, devoted to particular weapons systems or accounting systems. Most of the specific models for missiles such as Standard Arm, Shrike, Sidewinder and Phoenix were implemented on the HP 9830A. Several generalized models exist, mainly various forms of system effectiveness models for missiles. Predominant among these are ALWSS (Air-Launched Weapon Status Summary) and ALWSS2, which are dynamic, deterministic models programmed in FORTRAN IV and implemented on an IBM 7094 computer. Several models are devoted to work force and budgetary
analysis, such as Program Adjust which seeks to analyze the personnel composition of various FWED work teams, and compare it with "model" teams. (6)

In this chapter, the basic concepts of networks and their graphs were presented along with the fundamental concepts of simulation. Two types of network graphs and their associated mathematical representations were described. Due to the limited memory available on the HP 9830A neither of the mathematical models presented were economical enough in terms of memory utilization. As a result, an original mathematical model was developed for GINS. This model is described in the next chapter.
CHAPTER 3: THE GENERALIZED INTERACTIVE NETWORK SIMULATOR (GINS)

The purpose of this chapter is to describe the Generalized Interactive Network Simulator (GINS). This description will be divided into two sections. The first section describes the mathematical model developed for GINS while the second section describes in general the computer implementation of GINS.

GINS is a general, dynamic, deterministic, discrete model of a directed network graph. This model is predictive rather than normative (self-optimizing). GINS assumes that the network graph is a directed graph with unique ordinal numbers assigned to each link and node. An example of such a network is shown in figure 8, where nodes are designated by N1-N6 and links are designated by L1-L9.

Figure 8: Example network

The network is represented by an array of \( i \) rows by two columns, where \( i \) is the number of links, the first column...
contains the assigned initial node number for the link, and the second column contains the assigned terminal node number for the link. For the network of figure 8, the network array is shown in figure 9.

<table>
<thead>
<tr>
<th>Initial Node</th>
<th>Terminal Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>1</td>
</tr>
<tr>
<td>Link 2</td>
<td>2</td>
</tr>
<tr>
<td>Link 3</td>
<td>2</td>
</tr>
<tr>
<td>Link 4</td>
<td>2</td>
</tr>
<tr>
<td>Link 5</td>
<td>1</td>
</tr>
<tr>
<td>Link 6</td>
<td>4</td>
</tr>
<tr>
<td>Link 7</td>
<td>4</td>
</tr>
<tr>
<td>Link 8</td>
<td>3</td>
</tr>
<tr>
<td>Link 9</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 9: Matrix R

The advantage of this type of array is that it is most efficient in its utilization of array space. All of the array is used, rather than only part as is the case in the arrays of chapter 2. This property becomes important in any computer implementation in which memory is limited. This array will be referred to by the matrix R and is of dimension 1 by 2, where 1 is the number of links in the network.

The flow of entities in GINS is represented by a set of equations describing the link values as functions of
time, other link values (present or previous) or previous node values. These equations can be nonlinear equations or certain forms of differential equations. We will define a matrix $P$ to represent the flow values calculated by the set of independent flow equations. This matrix is dimensioned $m$ by $l$, where $m$ is any arbitrary integer greater than one and $l$ is the number of links in the graph. Likewise, we will define another matrix $S$ to represent the node values and dimension it $m$ by $n$, where $n$ is the number of nodes in the graph and $m$ remains as defined earlier. For the graph in figure 8, a possible set of flow equations would be:

\[
\begin{align*}
    P(t_1,1) &= 10P(t_1,5) \\
    P(t_1,2) &= 0.25S(t_2,2) - 0.10S(t_2,3) \\
    P(t_1,3) &= 0.50S(t_2,2) \\
    P(t_1,4) &= 0.25S(t_2,2) - 0.15P(t_1,5) \\
    P(t_1,5) &= 30(1 - e^{-(0.5t)}) \\
    P(t_1,6) &= 0.60S(t_2,4) \\
    P(t_1,7) &= 0.40S(t_2,4) - (P(t_1,3) - P(t_2,3))0.55 \\
    P(t_1,8) &= 0.90S(t_2,3) \\
    P(t_1,9) &= 0.85S(t_2,5)
\end{align*}
\]

In the above equations, $t_1$ represents the current time, while $t_2$ is $t_1-1$.

With the values of the links calculated, the node values are calculated according to equation (2):

\[
S(t_1,i) = S(t_2,i) - \sum_{j=1}^{l} P(t_1,j) + \sum_{j=1}^{l} PT(t_1,j)
\]

which states that any node value, is equal to the previous node value, minus the current link values for links.
beginning at that node (PT), plus the current link values for links terminating at that node (PT). The values of PI and PT are obtained from R. Equations (3) show equation (2) with PI and PT replaced by R:

\[ S(t_1, R(j, l)) = S(t_2, R(j, l)) - P(t_1, j) \]
\[ S(t_1, R(j, 2)) = S(t_2, R(j, 2)) + P(t_1, j) \]

where \( j = 1, 2, 3 \ldots 1 \)

Equations (3) imply conservation of flow, which is in fact observed in GINS. Simply put, conservation of flow means that node values are equal to the total flow into minus the total flow out of each node.

While equations (3) are accurate for continuous flow, they are not accurate for discrete flow. Since GINS is a discrete model, further refinements of equations (3) is necessary. Therefore, we define an operator \( T \), which truncates the decimal portion of a real number. Since we want only whole units of flow in GINS, we must replace the link values calculated by the set of equations (1) (which are real numbers), with integers obtained as follows:

let \( x_1 = P(t_2, j) - T(P(t_2, j)) \)
and \( x_2 = T(P(t_1, j) + x_1) \)

where \( P(t_2, j) \) is the previous flow value, and \( P(t_1, j) \) is the current flow value. We can restate equations (3) as follows:

\[ S(t_1, R(j, 1)) = S(t_2, R(j, 1)) - x_2 \]
\[ S(t_1, R(j, 2)) = S(t_2, R(j, 2)) + x_2 \]
In addition to altering the calculation of node values, we must also restate conservation of flow as the conservation of entities. We can state the conservation of entities as:

\[
\sum_{j=1}^{n} S(t,j) = K
\]

which says that the sum of the node values at any time \( t \) is a constant. If all initial state values are set to 0 and negative state values are allowed, \( K \) will be 0.

For the network in figure 8, the link and node values were calculated according to equations (1) and (4) for five time periods, assuming initial conditions of 0, and are displayed in table 1. As you can see, the sum of the values of each row in Table 1 on page 21 is 0 (since all initial conditions are 0), thereby illustrating conservation of entities.

One final aspect of GINS remains to be discussed; the role of constraints. Both node and link values have upper and lower constraints associated with them. All nodes and/or links can be limited by the same constraints or they can have unique constraints. If a node value ever exceeds these values, the model execution may be terminated. As with the nodes, each link has an upper and lower constraint which the link value must not exceed. However, in contrast to nodes, if a link exceeds a
Table 1
Results from figure 8
Table 1 (cont.)

<table>
<thead>
<tr>
<th>T</th>
<th>S 1</th>
<th>S 2</th>
<th>S 3</th>
<th>S 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-129</td>
<td>112</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>-337</td>
<td>194</td>
<td>35</td>
<td>77</td>
</tr>
<tr>
<td>3</td>
<td>-594</td>
<td>240</td>
<td>95</td>
<td>91</td>
</tr>
<tr>
<td>4</td>
<td>-880</td>
<td>273</td>
<td>114</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>-1182</td>
<td>290</td>
<td>126</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>-1496</td>
<td>302</td>
<td>132</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>-1816</td>
<td>309</td>
<td>137</td>
<td>103</td>
</tr>
<tr>
<td>8</td>
<td>-2140</td>
<td>314</td>
<td>138</td>
<td>104</td>
</tr>
<tr>
<td>9</td>
<td>-2466</td>
<td>314</td>
<td>141</td>
<td>106</td>
</tr>
<tr>
<td>10</td>
<td>-2794</td>
<td>317</td>
<td>142</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>S 5</th>
<th>S 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>161</td>
<td>237</td>
</tr>
<tr>
<td>5</td>
<td>199</td>
<td>476</td>
</tr>
<tr>
<td>6</td>
<td>209</td>
<td>752</td>
</tr>
<tr>
<td>7</td>
<td>220</td>
<td>1047</td>
</tr>
<tr>
<td>8</td>
<td>226</td>
<td>1358</td>
</tr>
<tr>
<td>9</td>
<td>231</td>
<td>1674</td>
</tr>
<tr>
<td>10</td>
<td>234</td>
<td>1998</td>
</tr>
</tbody>
</table>
constraint, the link may be forced to that value (overiding equations (1)) and model execution continued.

The GINS model is implemented in BASIC on a Hewlett Packard 9830A programmable calculator. The program is written to take advantage of the interactive capabilities of the calculator, which include communication in English via alphanumeric strings, a keyboard, an alphanumeric display and thermal printer. In addition to the calculator and printer, a peripheral cassette drive and an x-y plotter are available for use. A typical user of GINS will supply one or more blank cassette tapes which he inserts in the peripheral tape drive. The user then inserts the program tape and loads the first program file into the calculator. The calculator then proceeds to prompt the operator for the necessary model data such as number of nodes and links, length of simulation run, network configuration, flow equations, and constraints. This information is stored at the beginning of the first data tape for editing and rerun use. The program then performs the simulation, storing the node and link values on the data tape. After the values have been calculated, the program asks the operator which values he wants printed (if any) and which ones should be plotted. Those values indicated by the operator are then printed and/or plotted. The simulation may then be rerun in the same form, or the model may be changed before another run is
made.

The maximum number of nodes and links that can be accommodated by GINS is 100. The length of the simulation is unlimited, since the results are calculated in time segments of five or ten and stored on tape. However, as the size of the model and length of the simulation run increase, the time to perform the simulation correspondingly increases, due in part to increased tape manipulation.

Whenever a link constraint is violated, one of three actions takes place. One possible action is for the simulation to halt immediately after printing a notice of a constraint violation and its source. Another alternative is to print a notice of a constraint violation but continue with the simulation run. The final alternative is to force the link to the constraint value and continue the simulation. The alternatives for a node constraint violation are the same as for a link violation, with the exception that the final option can't be used, since this would violate the conservation of entities principle.

The prime difficulty encountered in implementing GINS was the limited core memory available in the HP 9830A. The maximum available memory was 7904 words, each word being sixteen bits long. The various data arrays alone required over 4000 words, while a maximum of 2000 words
was allotted for flow equations. This left a maximum of 2000 words for the program. Obviously, this was not enough, so the program was broken into 13 modules which are stored on a cassette tape called the program tape. During program execution, the various modules are brought into the calculator memory in the required sequence, as determined from the operator responses to prompts from the calculator display. With this technique, the program was squeezed into the calculator, but at the price of greater running time. If the entire program and more of the data could remain in the main memory, the speed of execution would be faster.
CHAPTER 4: VALIDATION & APPLICATIONS

The purpose of this chapter is to describe the methods used in the validation of GINS. In addition, the chapter discusses various applications of GINS and the modeling techniques behind those applications.

The validation of GINS was performed by running two models, one a logistics model, the other a digital network model. From these simulation runs, three basic tests were accomplished; the reproduction of past data, reasonableness, and completeness. The logistics model involved reproducing a model previously programmed in FORTRAN for a batch processor. The results of this model were available for comparison with the results of a GINS run of the model. The comparison of these results comprised the reproduction of past data test and the reasonableness test. The models and the results are discussed in the following paragraphs.

The logistics model, called the Air Launched Weapons System Status (ALWSS), is shown in figure 10. The GINS representation of the ALWSS model is shown in figure 11. Note the nearly one-for-one correspondence between the two models. In developing the GINS model from the ALWSS model, the numbered reservoirs were replaced by nodes, except for reservoirs #2, #8, and #10. The reservoir #2 (minimum reserve), which in fact is a subset of reservoir #1, was replaced in GINS by a constraint on the minimum
Figure 10: ALWSS
Figure 11: GINS representation of ALWSS
value of node 1 (Ready For Issue). Reservoir #8 was
combined with reservoirs #9 and #10 of the ALWSS model to
form node 7 of the GINS model. Figure 12 cross references
the ALWSS reservoirs with the GINS nodes.

<table>
<thead>
<tr>
<th>ALWSS reservoir</th>
<th>GINS node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>N1</td>
<td>Ready For Issue (RFI)</td>
</tr>
<tr>
<td>#2</td>
<td>N2</td>
<td>Aircraft</td>
</tr>
<tr>
<td>#3</td>
<td>N3</td>
<td>Captive flight</td>
</tr>
<tr>
<td>#4</td>
<td>N4</td>
<td>Ready For Launch (RFL)</td>
</tr>
<tr>
<td>#5</td>
<td>N5</td>
<td>Free flight</td>
</tr>
<tr>
<td>#6</td>
<td>N6</td>
<td>Fuzing</td>
</tr>
<tr>
<td>#7</td>
<td>N7</td>
<td>Probable success</td>
</tr>
<tr>
<td>#8</td>
<td>N8</td>
<td>Failure</td>
</tr>
<tr>
<td>#9</td>
<td>N9</td>
<td>Repair</td>
</tr>
<tr>
<td>#10</td>
<td>N10</td>
<td>Modification</td>
</tr>
<tr>
<td>#11</td>
<td>N11</td>
<td>Obsolete</td>
</tr>
<tr>
<td>#12</td>
<td>N12</td>
<td>Production</td>
</tr>
</tbody>
</table>

Figure 12: Network states

The "pipes" in the ALWSS model correspond to the
links in the GINS model. Figure 13 lists the GINS links
along with their interpretation obtained from figure 10.

The ALWSS model accepts as inputs the desired kill
rate, the procurement plan, and the modification plan and
then calculates the remaining rates based on the
reliability coefficients. When the quantity of rounds in
RFI dips below the minimum reserve, the model is
terminated. The method used with GINS was to input the
flow equations, with L7, L16 and L15 as constants or
functions and the reliability coefficients as constants.
Figure 14 shows the flow equations used in the GINS model. The model was run with the Nl lower constraint mode set to 1.

<table>
<thead>
<tr>
<th>GINS Links</th>
<th>ALWSS Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Onload rate (Q0)</td>
</tr>
<tr>
<td>L2</td>
<td>Captive flight rate</td>
</tr>
<tr>
<td>L3</td>
<td>Non-tactical mission rate (Qn)</td>
</tr>
<tr>
<td>L4</td>
<td>Firing rate (Qf)</td>
</tr>
<tr>
<td>L5</td>
<td>Kill rate (Qk)</td>
</tr>
<tr>
<td>L6</td>
<td>Free flight failure rate</td>
</tr>
<tr>
<td>L7</td>
<td>Warhead failure rate</td>
</tr>
<tr>
<td>L8</td>
<td>Pre-flight failure rate</td>
</tr>
<tr>
<td>L9</td>
<td>Captive flight failure rate</td>
</tr>
<tr>
<td>L10</td>
<td>Design life</td>
</tr>
<tr>
<td>L11</td>
<td>Modification plan</td>
</tr>
<tr>
<td>L12</td>
<td>Repair rate</td>
</tr>
<tr>
<td>L13</td>
<td>Modification rate</td>
</tr>
<tr>
<td>L14</td>
<td>Procurement plan</td>
</tr>
</tbody>
</table>

Figure 13: Network flows

This would cause the program to halt as soon as the level of RFI went below the minimum reserve, and the value of Nl and the time at which the constraint was exceeded were output. This allowed a direct comparison of the ALWSS and GINS results based on the time at which RFI went below the designated minimum reserve (referred to as "depletion"). Several runs were made on GINS, with differing firing rates and delayed firing schedules, and the results compared. The results are listed in figure 15:
10 DEF FN1(X)
20 P[T1, 1]=(P[T1, 5]+P[T1, 4])/(0.99*0.79)
30 P[T1, 2]=P[T1, 1]+0.97
40 P[T1, 3]=P[T1, 2]*0.99
50 P[T1, 4]=5
60 P[T1, 5]=P[T1, 7]/(1+0.79)
70 P[T1, 6]=P[T1, 7]/1
74 P[T1, 7]=0
76 IF T <= 12 THEN 90
80 P[T1, 7]=20
90 P[T1, 8]=P[T1, 5]*(1-0.79)
100 P[T1, 9]=P[T1, 6]*(1-1)
110 P[T1, 10]=P[T1, 1]*(1-0.97)
120 P[T1, 11]=P[T1, 2]*(1-0.99)
130 P[T1, 12]=5*(T2-9)+0
140 P[T1, 13]=0
150 P[T1, 14]=(P[T1, 10]+P[T1, 11]-P[T1, 13])=1
160 P[T1, 15]=15
161 IF S[T2, 10] >= 15 THEN 170
162 IF S[T2, 10] >= 0 THEN 165
163 P[T1, 15]=0
164 GOTO 170
165 P[T1, 16]=S[T2, 10]
170 P[T1, 16]=0
171 IF T<18 OR T>31 THEN 180
172 IF T#18 THEN 175
173 P[T1, 16]=9
174 GOTO 180
175 IF T#19 THEN 178
176 P[T1, 16]=40
177 GOTO 180
178 P[T1, 16]=46
180 RETURN X
190 END

Figure 14: Flow equations
<table>
<thead>
<tr>
<th>Run</th>
<th>ALWSS Depletion Time (Yr.)</th>
<th>GINS Depletion Time (Yr.)</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.65</td>
<td>0.67</td>
<td>+ 3 %</td>
</tr>
<tr>
<td>2</td>
<td>1.30</td>
<td>1.17</td>
<td>- 10 %</td>
</tr>
<tr>
<td>3</td>
<td>1.24</td>
<td>1.17</td>
<td>- 6 %</td>
</tr>
<tr>
<td>4</td>
<td>3.70</td>
<td>3.17</td>
<td>- 14 %</td>
</tr>
</tbody>
</table>

Figure 15: Comparison of results of GINS with ALWSS

The variation in the depletion times could have resulted from the incomplete documentation of the ALWSS runs, which could have lead to improper modeling in the GINS runs.

The fourth run of figure 15 was also plotted in another GINS run and the result compared with the ALWSS plot. Figure 16 shows the ALWSS plot and figure 17 shows the GINS plot. The major discrepancies between the plots are due to lack of documentation in the ALWSS report concerning the handling of modifications and new production. Nevertheless, the GINS results were reasonably close to the ALWSS results.

The second model run on GINS was a combinational logic circuit for performing addition. The problem was actually a trivial one in that all that was desired was a verification of the truth table for the circuit. However, the problem does show the use of logical operators in the flow equations and shows a technique for converting a static problem (verification of the truth table) into a dynamic one.
Figure 16: ALWSS plot
The logic circuit is shown in figure 18, in block representation.

Figure 18: Full adder

The inputs to the circuit are the carry bit from a previous stage (C1) and the two input digits (X1 and Y1). The output is the sum of the inputs (Si) and a carry bit for the sum (C1+1), which becomes an input to the next stage. The logic elements are all nand gates. The GINS network representation is exceedingly easy to generate. All that is needed is to redraw the logic circuit, replacing the nand gates with nodes, drawing individual links for multi-connected outputs and inputs, and supplying source and sink nodes for the overall inputs and outputs. The GINS network for figure 18 is shown in figure 19.
To simulate the logic elements, the GINS flow equations used a NOT command and an AND command. All flows in the model are instantaneously dependent on the flows leaving the source nodes. These flows were entered as tabular data. The set of flow equations is given in figure 20. The P's in the flow equations correspond to the link numbers in figure 19.

To compare the results of the GINS simulation with the real circuit, the truth table for the adder was compared with the behavior of L19 and L20 when all possible input values were assumed as functions of time. Figure 21 shows the truth table for the full adder of figure 18.
10 DEF FN1(X)
20 P[T1,2]=P[T1,1]
30 P[T1,4]=P[T1,3]
40 P[T1,6]=P[T1,5]
50 P[T1,7]=P[T1,8]=P[T1,9]= NOT (P[T1,4] AND P[T1,5])
60 P[T1,10]= NOT (P[T1,3] AND P[T1,7])
70 P[T1,11]= NOT (P[T1,6] AND P[T1,8])
80 P[T1,12]=P[T1,13]= NOT (P[T1,10] AND P[T1,11])
90 P[T1,14]=P[T1,15]=P[T1,16]= NOT (P[T1,2] AND P[T1,12])
100 P[T1,17]= NOT (P[T1,1] AND P[T1,14])
110 P[T1,18]= NOT (P[T1,15] AND P[T1,13])
120 P[T1,19]= NOT (P[T1,17] AND P[T1,18])
130 P[T1,20]= NOT (P[T1,9] AND P[T1,16])
140 RETURN X
150 END

Figure 20: Flow equations

<table>
<thead>
<tr>
<th>C1</th>
<th>X1</th>
<th>Y1</th>
<th>S1</th>
<th>C1+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 21: Truth table for full adder

A sequence of input values was tabulated for L1(C1), L3(X1), L5(Y1) to duplicate the values for C1, X1 and Y1 in figure 21 and preloaded in the data files prior to the GINS run. Each value was held for a total of five time increments. The results of the run, and the inputs, are presented on the following pages.

Upon comparison of figure 21 and these tables, it becomes obvious that the GINS simulation produced the same
GINS REPORT
FULL ADDER SIMULATION
SYSTEMS ANALYSIS BRANCH
CODE 2243
JUNE 22, 1976

<table>
<thead>
<tr>
<th>T</th>
<th>P1</th>
<th>P3</th>
<th>P5</th>
<th>P19</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2
<table>
<thead>
<tr>
<th>T</th>
<th>P 1</th>
<th>P 3</th>
<th>P 5</th>
<th>P 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2 (cont.)
**GINS REPORT**

**FULL ADDER SIMULATION**

**SYSTEMS ANALYSIS BRANCH**

**CODE 2243**

**JUNE 22, 1976**

<table>
<thead>
<tr>
<th>T</th>
<th>P1</th>
<th>P3</th>
<th>P5</th>
<th>P20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3
<table>
<thead>
<tr>
<th>T</th>
<th>P 1</th>
<th>P 3</th>
<th>P 5</th>
<th>P 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>47</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3 (cont.)
results as those in figure 21.
CHAPTER 5: CONCLUSION

Based on the results of the problems described in Chapter 4, it is felt that GINS meets the objective set at the outset of the project. That objective is to provide Systems Analysis Branch with a general modeling capability using existing equipment (the Hewlett Packard 9830A). The ALWSS model described in Chapter 4 is typical of the anticipated applications of GINS. In particular, GINS will most likely be used by Systems Analysis Branch to certify the results of a much more complex model called the System Effectiveness Model (SEM). This will be done by checking the results of both models for consistancy and reasonableness. However, as the second example of Chapter 4 indicates, GINS may be applied to problems beyond the realm of logistics. The range of problems that can be handled by GINS is limited by the ability of the user to resolve the problem into a network-flow graph, and the amount of data that must be processed at each time increment. The anticipated advantages of GINS have been realized, such as fast response (turn-around time) and interactive operation. Specifically, the turn-around time for a typical problem (for example, ALWSS) is on the order of an hour, compared to a day for a batch process run. This time could be reduced significantly if the peripheral cassette drive could be speeded up or replaced by more accessible memory. The structure of the GINS
program allows considerable interaction between the operator and the model, and makes operation of the program quite simple. As implied above, the only difficult task facing the GINS user is to formulate a network flow model for the problem.
REFERENCES

(1) Connor, John A.; Systems Effectiveness Role in Fleet Weapons Engineering Directorate

(2) Connor, John A.; Management Decision Applications of System Effectiveness Models; November 9, 1972

(3) Frank, Howard and Ivan T. Frisch; Communication, Transmission, and Transportation Networks; Addison-Wesley; 1971

(4) Woodgate, H. S.; Planning by Network; Brandon/Systems Press, New York; 1967

(5) Fishman, George S.; Concepts and Methods in Discrete Event Digital Simulation; John Wiley & Sons, New York; 1973

(6) BACADE Annual Project Report For Fiscal Year 1975

(7) Cox, J. Grady and Joe H. Mize; Essentials of Simulation; Prentice-Hall, Inc.; Englewood Cliffs, New Jersey; 1968
APPENDIX A: SYSTEM IMAGE & DATA MANIPULATION

The system image of GINS is comprised of five arrays and two vectors, the P, S, T, U, C, R and Z arrays. These arrays contain the network data necessary for a GINS run, and are filled during the input routine. The only network information not contained in these arrays is the set of flow equations. These equations are contained in a special function key defined as function FNA. The results of the simulation run are also stored in these arrays, specifically the P and S arrays.

The P array is a split-precision array, initially dimensioned to 6 by 100, which contains the link values as calculated by the equations in FNA. After the size of the network and the run length of the simulation have been input to the machine, the P array is redimensioned to 1 rows by n columns, where 1 is the number of time intervals per data file and n is the number of links (see figure 22). In all cases, 1 is either 6 or 11, and n is less than 100. Since the run length is unlimited, the link values are stored on a cassette tape in blocks of 5 or 10, depending on the network size. The first block of link values is predefined to pi in order to allow FNA to be executed, then FNA is repeatedly executed until the link values no longer change. If the flow in some links is defined by tabular data rather than by the equations in FNA, then these values are entered prior to processing.
The S array is also a split-precision array of initial dimension 6 by 100, which contains the node values derived from the link values in the P array. The S array is redimensioned to the number of nodes in the problem network (see figure 22). As with the P array, the number of rows in S is either 6 or 11, depending on the network size. When the first 5 or 10 sets of node values have been calculated, they are stored in a file on the data tape. Except for the number of columns and the predefinition exercise, the P array and the S array are identical.

The T array is a split-precision array of 3 rows by 100 columns which holds the initial condition and constraint data for the P (link) array (see figure 22). The first row contains the initial conditions for the links. The second and third rows contain the upper and lower link constraint values for each link. This array is loaded during the input routine and is immediately stored on the beginning of the first data tape. During the output routines, this array is used to store special output flags used to determine which links are to be output.

The U array is another split-precision array of 3 rows by 100 columns which holds the S (node) array initial condition and constraint data. The data is arranged and manipulated in the same manner as the T array. The P, S,
The \( T \) and \( U \) arrays are shown in figure 22. In figure 22, \( l \) is the number of links in the network, \( n \) is the number of nodes, and \( Z(8) \) is 5 or 10.

The \( C \) array is an integer array of initial dimension 100 rows by 2 columns and contains the incidence matrix for the network. This array is filled during the input routine by operator responses to prompts from the calculator display. The rows correspond to the network links and are arranged in ascending numerical order. The first column contains the beginning node number for the link, and the second column contains the terminating node number for the link. Figure 23 shows the \( C \) array configuration. The \( C \) array is identical to the incidence matrix \( R \) referred to in Chapter 3, p.16.

The \( R \) vector is an integer vector which contains the constraint mode information for the network. This information tells the program which constraint mode to assume and is given in the form of a 4-digit number. The first two digits refer to the upper and lower link constraint mode respectively, and the last two digits refer to the upper and lower node constraint mode respectively. Figure 24 shows an example code number and the three mode numbers.
Figure 22: P, S, T and U arrays

Figure 23: The C array
The Z vector is an integer vector of 15 elements which contains counters and flags used to control the execution of the program. Figure 25 lists the elements of Z and describes their functions:

| Z(1) | - data tape # |
| Z(2) | - last tape flag |
| Z(3) | - # of used files per tape |
| Z(4) | - cumulative # of used files |
| Z(5) | - # of nodes in network |
| Z(6) | - # of links in network |
| Z(7) | - # of time intervals in problem |
| Z(8) | - # of time intervals per file |
| Z(9) | - # of iterations of FNA per time increment |
| Z(10) | - rerun flag |
| Z(11) | - table flag |
| Z(12-15) | - unused |

In order to calculate current path and state values, the values of paths and states calculated just prior to the current values must be available. Since each row of the S and P arrays represent a single increment of time,
only a limited "time slice" of the entire run can be operated on at any time (due to core constraints). Each of these "time slices" is stored on tape when completed (see figure 26 step 2). However, to begin a new "time slice" the last values of the previous "time slice" must be available. If certain path values have been entered as tabular data prior to processing, then the next data file must be read from the data tape and placed in memory (figure 27, step 1). If only part of the S and P arrays could be read from tape, then all that would be needed to save the last values would be an extra row. This is not possible, so another array must be used to hold the last values. However, because of core limitations, a new array or vector cannot be used. Since the initial condition data is used only at the beginning of the run, this data can be saved on tape and the first rows of the T and U arrays can be used to hold last values during the remainder of the run (see figure 26 step 1). When a new "time slice" is started, the contents of the first row of the T and U arrays must be copied into the last row of the S and P arrays, so that last values throughout the run have the same array name (see figure 27 step 2).
At the start of a process run, additional data manipulations are performed to initialize the data arrays. The first of these manipulations occurs to the P array, and consists of the loading of the calculator memory with the contents of the first data file (see figure 28 step 1). This operation is performed on the very first pass of the first run in order to determine the number of iterations through the system of flow equations.
required to insure that all variables have been properly defined, and to identify tabular data. When the contents of the first path data file have been loaded, the calculator then copies the contents of the first row of the T array into the first and last rows of the P array (see figure 28 steps 2&3). This last operation is also performed on the S array (see figure 29 step 1). The data manipulations at the end of the first pass are identical to those of figure 26. During the process of determining the number of iterations through FNA, the first row of the T array is used once again as the last value storage vector.

During the report generator routine, the first row of the T and U arrays are used to store the output flags which indicate which states and which paths are to be printed. The flag consists of the value 2.9979 (the metric value of the speed of light * 100 million). Once again, this action does not destroy the initial condition values, since they are stored on the first data tape. The same process is used to flag states and paths for plotting. To insure that a state or path flagged for a printed report is not inadvertently plotted, the first row of the U and T arrays is set to 0 at the start of the plot routine.
As mentioned previously, the GINS program is stored on a cassette tape, as are the results of the simulation run. The structure of the GINS program tape is given in figure 30. It should be noted that files 2 and 3 perform the initial input function; files 5, 6, and 7 perform the edit function; and files 16, 17, and 18 perform the graphing function. File 9 is an unused and untested subroutine which is designed to delete nodes and links from the network.
<table>
<thead>
<tr>
<th>File #</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Empty</td>
</tr>
<tr>
<td>1</td>
<td>Main program</td>
</tr>
<tr>
<td>2</td>
<td>First input subroutine</td>
</tr>
<tr>
<td>3</td>
<td>Second input subroutine</td>
</tr>
<tr>
<td>4</td>
<td>Empty</td>
</tr>
<tr>
<td>5</td>
<td>Edit-change subroutine</td>
</tr>
<tr>
<td>6</td>
<td>First edit-add subroutine</td>
</tr>
<tr>
<td>7</td>
<td>Second edit-add subroutine</td>
</tr>
<tr>
<td>8</td>
<td>Empty</td>
</tr>
<tr>
<td>9</td>
<td>Edit-delete subroutine (not used)</td>
</tr>
<tr>
<td>10</td>
<td>Empty</td>
</tr>
<tr>
<td>11</td>
<td>Tabular input subroutine</td>
</tr>
<tr>
<td>12</td>
<td>Processor subroutine</td>
</tr>
<tr>
<td>13</td>
<td>Empty</td>
</tr>
<tr>
<td>14</td>
<td>Tabular report subroutine</td>
</tr>
<tr>
<td>15</td>
<td>Empty</td>
</tr>
<tr>
<td>16</td>
<td>Axis subroutine</td>
</tr>
<tr>
<td>17</td>
<td>Data plot subroutine</td>
</tr>
<tr>
<td>18</td>
<td>Label subroutine</td>
</tr>
</tbody>
</table>

**Figure 30: Program Tape structure**

The first data tape differs from subsequent data tapes in that it contains the network and network parameter data in addition to the node and link values obtained during processing. The structure of the first data tape is given in figure 31. All other data tapes have a common structure which differs from the structure of the first data tape in that the network data in files 1-5 is absent. Therefore, for data tapes 2-K, file 0 contains the Z array, and files 2-31 contain the calculated node and link values.
<table>
<thead>
<tr>
<th>File #</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tape ID, flags, &amp; counters (Z array)</td>
</tr>
<tr>
<td>1</td>
<td>Link IC &amp; constraints (T array)</td>
</tr>
<tr>
<td>2</td>
<td>Node IC &amp; constraints (U array)</td>
</tr>
<tr>
<td>3</td>
<td>Constraint modes (R vector)</td>
</tr>
<tr>
<td>4</td>
<td>Network structure (C array)</td>
</tr>
<tr>
<td>5</td>
<td>Flow equations (special function key 1)</td>
</tr>
<tr>
<td>6,8...30</td>
<td>Link values from processing</td>
</tr>
<tr>
<td>7,9...31</td>
<td>Node values from processing</td>
</tr>
</tbody>
</table>

Figure 31: First Data Tape structure
APPENDIX B: GINS OPERATING MANUAL

Section 1 - Path Equations & Tabular Data

GINS is a network simulation program implemented on a Hewlett Packard 9830A programmable calculator. To use GINS, the operator must have resolved his problem into a network flow graph as shown below:

![Network Flow Graph]

Figure 32: A network flow graph

Each state is assigned an integer number, beginning with one. Each path is also assigned an integer number beginning with one. A set of equations must be supplied which describes the flow of entities between states, each path having a defining equation. The format of these equations is illustrated in figure 33:

\[ P(T1,1) = 0.5 \times T \times S(T2,2) \]

Figure 33: Example flow equation
The P stands for path, the S stands for state, T1 represents the current variable pointer, T2 represents the previous variable pointer, and T represents the current value of time. The number inside the parentheses occurring after the comma is the state or path designator, and corresponds to the numbers in the network flow graph. Therefore, the flow equation in figure 33 states that the flow in path number 1 (in figure 32) is equal to half the current value of state number 2. Two important restrictions must be followed when writing the flow equations: a T2 must never appear on the left-hand side of an equation; and a T1 must never be used in conjunction with an S. The reason for the first restriction is that all flow equations must define current flow values only. The reason for the second restriction is that current state values are calculated from current flow values, and are undefined at the time the current flow values are being calculated.

In addition to arithmetic functions, several other functions are available for use in the flow equations. Please refer to the first two chapters of the calculator handbook. Unless the operator has worked with this machine before, these chapters should be read.

As an alternative to path equations, the operator may specify path values by tabular data. This data is entered before the other paths and states are calculated. Tabular
values should be used when path values cannot be described by equations or when the equations would be too complex for efficient use. Tabular data requires considerable time to input, and must be re-entered on each input or edit pass, therefore it is advantageous to use equations where possible.

In addition to supplying a network model and a set of path equations or values, the operator must have at least one cassette tape for storing the model configuration and result data. If more than one tape is needed due to the length of the simulation run, the operator will be informed by the calculator (see sections 3 & 4).
Section 2 - Operating Modes

In using GINS, the operator has a variety of modes which may be exercised. Comprising these modes are three input modes and two output modes. The input modes allow the operator to selectively change model parameters, input new models, or run an unchanged model repeatedly. The output modes include one for producing tables of numerical data and another for generating plots of that data.

To start using GINS, the operator must perform the following operations:

1. Switch on power to calculator, printer and peripheral cassette drive
2. Insert program tape in the internal cassette drive of the calculator
3. Insert first data tape in the peripheral cassette tape drive
4. Press the LOAD key found in the upper right corner of the keyboard
5. Press the 1 key
6. Press the EXECUTE key

The internal cassette drive will load the first program file into the calculator memory. The operator must now do the following:

1. Press the RUN key
2. Press the EXECUTE key

When this sequence of keystrokes has been completed, the calculator will display the following question:

INPUT?
If a new model is desired, the operator should type YES and press the EXECUTE key. This response will cause, the first input-mode file to be linked into the calculator memory and program execution will continue (see Section 3 for further details).

If a new model is not to be run, the operator should type NO and press the EXECUTE key. This response will be followed by the message:

**INSERT FIRST TAPE--TYPE GO?**

If the first data tape is not in the external cassette drive, it should be placed there, followed by typing GO and pressing the EXECUTE key.

The calculator will then respond with:

**FIRST RUN?**

If the calculator has just been turned on or the RUN-EXECUTE key sequence has been given to start the program, then the response should be YES-EXECUTE (note: all responses from the keyboard must be ended by pressing the EXECUTE key). If the current pass was not started by a RUN-EXECUTE sequence, then the response should be NO. In either case, the calculator will return with the prompt:

**RERUN OR EDIT?**

If it is desired to run the last model configuration with no changes, then the operator should type RERUN. If changes in the model parameters are desired, the correct
response is to type EDIT.

When the model processing has been completed, the calculator will display the following prompt:

REPORT?

If a table of results is desired, the response YES will link in a subroutine which generates tabular results. The operator may select any or all of the paths and states as output. If a NO is typed in response (or when the report has been completed), the calculator will display:

PLOT?

If a graph of the result data is desired, a YES response will cause the first plot subroutine to be linked into memory. The plot routines allow the operator to specify one or more paths or states to be plotted and labeled. The operator may choose any label or title for the plots. If a NO response is keyed in, or when the plot subroutine is finished, the calculator will display:

FINISHED?

If another simulation run is desired, a NO response should be entered. This response will cause program execution to loop back to the INPUT prompt and start another simulation run. If a YES response is keyed in, the program will store all model parameters and then terminate.
Section 3 - Input Subroutine

The input subroutine is that part of GINS which initializes and enters all parameters concerning the network model. These parameters include the number of states and paths, the simulation run length, constraints, and initial conditions. In addition, the input subroutine marks the data tape(s), places identifiers on them, and stores all model parameters on the first data tape.

After the operator has responded to the INPUT prompt with a YES, the calculator displays:

HOW MANY NODES?
The operator must now key in the number of states in his network model, followed as always by the EXECUTE key. The calculator will then display:

HOW MANY LINKS?
The operator then keys in the number of paths in the model. The calculator will then prompt:

HOW MANY TIME INTERVALS?
The operator then enters the number of time intervals in the simulation run. If the run length requires more than one data tape, the calculator displays the message:

TOO MUCH DATA
followed by:

GET MORE TAPES, TYPE GO?
When the operator is ready to proceed, the word GO is typed. The calculator will then begin marking the data
tapes. This process requires about 10 minutes per tape. When each tape has been marked, a prompt will be displayed requesting insertion of another data tape if needed.

After all the data tapes have been marked, the second input subroutine file is linked into the calculator memory, and the calculator displays the question:

LINK INITIAL CONDITION ALL?

If all paths have the same initial value at time 0, then a YES response is typed. The calculator will then display:

VALUE?

The initial condition value is then keyed in (along with the EXECUTE key). If the initial conditions are not all the same, then for each link, the message is displayed (after a NO response):

LINK (X)?

where X is the particular link number. The operator then keys in the initial condition of the link(X). After the initial conditions have been entered, the calculator displays the prompt:

UPPER LINK CONSTRAINT ALL?

If all the links are to be constrained to the same maximum value, then a YES response is entered. The calculator then displays the same prompt as described in the initial condition example above, and the same response is made by the operator. If the maximum link constraints are not the same, the same procedure as described for initial
conditions is followed.

Following the entry of upper link constraint data, the calculator displays:

LOWER LINK CONSTRAINT ALL?

Once again, the procedure is the same for inputting minimum link constraint values as for initial condition & maximum link constraint values.

When the link constraint and initial condition values have been entered, the same series of prompts is displayed for the nodes; and the same response procedure must be followed. When the node initial conditions & constraint values have been entered, the calculator displays:

CONSTRAINT MODE ALL?

The constraint mode defines how constraint violations will be handled by the processor. Three modes are available for path values, and two modes are available for state values. These modes are defined below:

1 - Upon a violation, the program jumps to the FINISHED prompt; another run may then be made, or execution is halted; the current run is terminated in either case, after the violating element is identified

2 - Upon violation, the offending element is constrained to the limit it has exceeded (for paths only)

3 - Upon a violation, the element in violation is identified but processing continues without any change in the offending element's value

The constraint modes are represented by the numbers given
above and are entered as a four digit number as shown below:

Four Digit Representation

1st Digit - upper path limit
2nd Digit - lower path limit
3rd Digit - upper state limit
4th Digit - lower state limit

Two Digit Representation

1st Digit - upper limit
2nd Digit - lower limit

The four digit representation is used in the input routine, while the two digit form is used in editing, in which case the context of the edit determines which category (path or state) is being used. A typical mode would be 2233.

If the four digit mode number is the same for all elements (nodes & links), then the operator keys in YES to the CONSTRAINT MODE ALL prompt. If the four digit mode number is not the same for all elements, then the operator types NO. The calculator then displays:

CONSTRAINT MODE (I)?

where I is a counter whose maximum value is the number of links in the model. The operator then enters the mode number that is desired for node & link I.

After the constraint mode data has been entered, the calculator displays:
LINK (I) FROM, TO, REP?
The operator then types the number of the node from which
link I begins, a comma, the number of the node at which
link I terminates, a comma, and a short description of the
link. For the network in figure 32 on page 56, the
response for link 1 would be 1, 2, "A PRODUCTION ". Note
that the description must be set off by quotation marks.
If no description is available or desired, a single
quotation mark followed by (as always) the EXECUTE key
should be used. In either case, the calculator then
prints:

LINK (I)-- "DESCRIPTION"

This process is repeated for each link.

When the network entry has been completed, the
calculator displays:

NODE (I) REPRESENTS?
The operator then types a short description of the node
and ends it by pressing the EXECUTE key. The calculator
then prints:

NODE (I)-- DESCRIPTION

This process is repeated for each node.

After all node descriptions have been printed, the
calculator displays:

ENTER FLOW EQUATIONS

This message is followed shortly (about 7 seconds later)
by:
SEE OPERATING MANUAL (CONT 1390)

which is both displayed and printed by the calculator. The operator will now start entering the flow equations described in Section 1. The procedure is:

1. Press the FETCH key (in the edit key group)
2. Press the f1 key (in the special function key group)
3. Press the EXECUTE key
4. Press the AUTO# key (in the program key group, number 10 will appear on the display)
5. Press the PRT ALL key
6. Type DEF FNA(X)
7. Press END OF LINE key (last display number plus 10 will appear on display)
8. Type in a link equation
9. Press END OF LINE key

Steps 8 and 9 are now repeated until all path equations have been entered. Each time the END OF LINE key is pressed, that line will be printed by the calculator.

When all flow equations have been entered, the operator then does the following:

1. Type RETURN X
2. Press END OF LINE key
3. Type END
4. Press END OF LINE key
5. Press PRT ALL key
6. Press END key
7. Press CONT key
8. Type 1390
9. Press EXECUTE key

The calculator will now save the flow equations on the data tape and link in either the tabular data subroutine or the processor subroutine. At this point, the input subroutine has been completed.
Section 4 - Edit Subroutines

The edit subroutines allow the operator to modify a previously entered model under program control. There are two edit subroutines, the change subroutine and the add subroutine. The change subroutine allows the operator to change node & link parameters and flow equations. The add subroutine allows the operator to increase the number of nodes & links in the model, or to increase or decrease the time length of the simulation run.

After the operator has entered a YES response to the EDIT OR RERUN prompt, the calculator displays:

CHANGE, ADD?

If the operator wishes to exercise the change mode, a C is typed, and if the add mode is desired, an A is typed.

Change Mode

If the change mode has been selected, the calculator will display:

LINK?

If the operator wishes to change any of the link parameters in the model, a YES response must be entered. The calculator will then display:

EDIT ALL?

If the change desired is to apply to all links then a YES response must be typed. If the desired change is to apply to only one link, then a NO response must be typed. If a NO is typed, the calculator will display:
WHICH LINK?
The operator then keys in the number of the link to be changed.

After the operator has indicated the scope of the edit, the calculator displays:

STRUCTURE?
If the network is to be rearranged, then a YES is typed. The calculator will then display:

LINK (I) FROM, TO?
As in the input subroutine, the operator must now enter the node number for the link I starting node, a comma, and the link I terminal node number. If the entire network is to be changed, this sequence will repeat until all links have been located. If the network is not to be rearranged or when the changes have been made, the calculator displays:

INITIAL CONDITION?
If changes in the link initial conditions are desired, a YES response is entered. The calculator then displays:

VALUE?
The new initial condition is then entered. The calculator will then display:

CONSTRAINT?
If any of the link constraints are to be changed, a YES is typed. The calculator then displays:

UPPER LIMIT?
The upper constraint value is now typed by the operator. The calculator will next display:

LOWER LIMIT?

Again, the operator enters the new lower limit value. The calculator will now display:

CONSTRAINT MODE?

If the link constraint mode is to be changed, then a YES is typed. The calculator then displays:

NEW MODE?

The two digit link constraint mode is now keyed in by the operator (see Section 3, pp.64-65 for constraint mode descriptions).

When all link parameters have been edited (or skipped), and if the links are being edited individually, the calculator displays:

ANOTHER LINK?

If another link's parameters are to be changed, a YES is typed. The calculator then displays WHICH LINK and the process is repeated. If no other changes to other links are desired, a NO is typed and the calculator displays:

ENTER FLOW EQUATIONS

After 7 seconds, the calculator displays and prints:

SEE OPERATING MANUAL (CONT 1690)

If no changes to the flow equations are desired, the operator does the following:
1. Press CONT key
2. Key in 1690
3. Press EXECUTE key

If changes in the flow equations are desired, the operator performs the following:

1. Press the FETCH key
2. Press the fl key
3. Press the EXECUTE key

The operator may now edit the flow equations using the edit keys (see calculator handbook chapter 2). If the operator needs a printed copy of the flow equations one can be generated by pressing the PRT ALL key. This will cause the printer to print the contents of fl. In editing the flow equations, the operator fetches the line number of the equation to be changed. The line will be displayed and the edit keys become operative on that line. When the flow equations have been entered, the calculator displays:

MORE EDITING?

if a YES is typed, the program loops back to the start of the edit change subroutine. If the answer is NO, then the program continues to the tabular data prompt.

Add Mode

If the add mode has been selected, the calculator displays:

HOW MANY NEW LINKS?

The operator then keys in the number of links being added
to the model. If no new links are being added, the number entered is 0. The calculator will next display:

HOW MANY NEW NODES?
The operator must key in the number of new nodes (0 if none) being added to the model. The calculator will then display:

HOW MANY TIME INTERVALS?
The operator must now input the number of time intervals in the model run. If the number of time intervals requires more data tapes, the calculator instructs the operator to get more tape. The calculator will then display:

INSERT LAST TAPE - TYPE GO?
The operator now inserts the final data tape (from the previous run) in the external cassette drive, and types GO. The calculator then determines if a new tape is required for marking. If this is the case, the calculator displays:

INSERT NEXT TAPE--TYPE GO?
Once again, the operator must insert the tape and type GO. The calculator will continue this process until all the new tapes are marked. When all tapes have been marked, the calculator displays:

INSERT TAPE (I)-TYPE GO?
The operator inserts the tape I and types GO. This process is repeated until all the old data tapes have been
updated with new file and time counts. The calculator then links in the second add file and displays:

LINK INITIAL CONDITION ALL?

From here on through the rest of the add subroutine, the prompts and operator responses follow the same format as the second input file (see pp.63-68, Section 3).

When the model parameters for the new nodes and links have been entered, the calculator displays:

FINISHED?

If more editing of the model is desired, the program loops back to the start of the edit subroutine.
Section 5 - Tabular Input Subroutine

The function of the tabular input subroutine is to provide the capability to use input data in tabular form as well as equations. If the operator answers YES to the TABULAR DATA prompt, the calculator will link in the tabular data subroutine file, and then displays:

INSERT TAPE K - TYPE GO?

where K is the tape number, and starts with 1. The operator then inserts the tape and types in GO. The calculator then displays:

FROM T X1 TO?

where X1 is the next time period for which data has not been entered (for the first pass, X1 is 0). The operator then keys in the last time period for which the data value exists. For example:

Display - FROM T2 TO?

Desire - single data point of 5 at T2 for link 1

Respond - a 2

the calculator will then display:

LINK?

the operator must now key in the link number to which the tabular data corresponds. For the above example, the operator keys in a 1. The calculator will then display:

VALUE?

The operator must enter the data value. For the example,
the operator keys in a 5. The calculator will then display:

FROM T3 TO?

and the process will repeat until the data file has been filled for that link. The calculator then displays:

MORE LINKS?

If tabular data is to be entered for other links, a YES response is made. The calculator repeats the previous process until the file is once again full. When all tabular data has been entered for a given file (or time slice), a NO is typed in response to the above prompt. The calculator then saves the data on the data tape and starts the process over for the next file.

When all data files on all data tapes have been filled with the desired tabular data, the calculator displays:

INSERT FIRST TAPE - TYPE GO?

The operator will insert the first data tape in the external tape drive and type GO. The calculator will then link in the processor file and begin the simulation run.
Section 6 - Tabular Report Subroutine

The purpose of the tabular report subroutine is to print tables of data generated by the processor subroutine. Any combination of states and paths may be output. The format of the reports is shown in table 1, p.20. During the printing process, all the desired paths are printed first, then all states are printed starting on a new line.

When the processor has completed its calculations, the calculator will display:

REPORT?

If the operator desires the tabular reports, a YES is typed. The calculator then links in the tabular report subroutine and displays:

TEAR OFF EXCESS PAPER

The operator then has seven seconds in which to advance and remove paper from the beginning of the report. Next, the calculator prints GINS REPORT and displays:

HEADING?

If a heading for the report is desired, a heading is typed on the keyboard. The calculator will then center and print the heading. If the operator does not desire a heading, pressing the space bar and the EXECUTE key will cause the printer to skip the heading and continue by printing SYSTEMS ANALYSIS BRANCH and CODE 2243. The calculator will then display:
DATE?
The operator may then type in the date in free format. The calculator will then print the date. If no date is desired, the typing of the space bar will cause the printer to skip the line.

In either case, the calculator will then display:

PRINT WHICH LINKS?
The operator then types in the numbers of the links to be printed, each number separated by a comma. If no links are to be printed, the operator should type NONE, which causes the program to skip the link printing. If all of the links are to be printed, the operator enters ALL. In any event, the calculator will next display:

PRINT WHICH NODES?
The operator must then respond in one of the modes described for the links above. After the operator has responded, the calculator will print a header for the first four paths and display:

INSERT TAPE K - TYPE GO?
The operator must now insert data tape K in the external tape drive and type GO.

The calculator now loads the data files and prints the first four selected paths. Whenever a data tape is exhausted, the calculator prompts the operator to insert another tape. When all the first four path values have been printed, the calculator prints a new header for the
next four paths and prompts the operator to insert the first data tape. This process repeats until all the selected paths have been printed.

After all the selected paths have been printed, the calculator repeats the process for the selected states. Because of the repeated passes made through the data tapes to print reports, the time to make a complete report may be considerable. The tabular report subroutine automatically pages the report, and each page may be removed from the printer when it is completed. The calculator waits four seconds between pages, and thus requires careful attention. If the operator misses a chance to remove a page, that page can be cut later. At no time during the printing process should the operator use the paper advance button on the printer. Remember, the operator's presence is required during the printing process to change data tapes.
Section 7 - Plot Subroutines

The function of the plot subroutines is to produce graphs of the data generated by the processor subroutine. The graphs can include plots of any state or path and as many as desired on a single graph. The operator has a choice of time scales for the x-axis and may select the range and label for the y-axis. The operator also has the option of labeling each plot and providing a title for the graph to be lettered by the plotter. The plot function is performed in three separate subroutines. These subroutines are establishing, drawing, and labeling the axes; plotting the data; and labeling the plots and the graph.

Prior to making a graph, the tabular data for the plots must be available, since the range of values must be known in specifying the y-axis range. When the tabular report subroutine is finished, the calculator will display:

PLOT?

If a graph is desired, a YES is typed and the calculator then links in the first plot subroutine, the axis subroutine.

Axis Subroutine

The calculator will now display:

PLACE PAPER ON PLOTTER

After four seconds, the calculator will display:

TYPE READY?
The calculator will now wait for the operator to ready the plotter for use. This procedure is described below:

1. Place plotting paper on plotting surface along bottom guide (move plot bar out of way by hand)
2. Press the LINE ON button (line light will illuminate)
3. Press the CHART HOLD button
4. Smooth paper with hand
5. Lower pen and press PEN UP button
6. Press LOWER LEFT button (pen will move to lower left quadrant of plotting surface)
7. Using the two knobs to the left of the Graph Limits buttons, move pen to the desired corner limit of the plotting area on the graph paper
8. Press the UPPER RIGHT button (pen will move to the upper right quadrant of plotting surface)
9. Using the two knobs located to the right of the Graph Limits keys, move pen to the desired corner of the plotting area on the graph paper
10. Check limits by pressing LOWER LEFT and UPPER RIGHT keys and observing the pen location

When the plotter is ready to begin drawing the graph, the operator types READY. The calculator will display:

```
INSERT TAPE 1 - TYPE GO?
```

The operator then places the first data tape in the external cassette drive and types GO. The calculator then displays:

```
TIME SCALE
```
After four seconds, the calculator then displays:

\[ \text{SECl,MIN2,HR3,DAY4,MO5,YR6?} \]

The operator then types the number associated with the desired time scale. The calculator then displays:

\[ \text{MAX Y VALUE?} \]

The operator then enters the maximum value of the paths or states that are to be graphed. The calculator then displays:

\[ \text{MIN Y VALUE?} \]

The operator then enters the minimum value of the paths or states that are to be graphed. The calculator then draws and labels the x-axis.

After the x-axis has been drawn and labeled, the calculator displays:

\[ \text{Y-LABEL?} \]

The operator then types whatever label is desired. If no label is desired, the operator types NONE and the default label will be UNITS. The calculator then draws the y-axis and labels it. The calculator now links in the second plot subroutine, the data plot subroutine.

Data Plot Subroutine

After the data-plot subroutine has been linked into the calculator memory, the calculator loads in the first data file and displays:

\[ \text{LINK?} \]

If the operator wishes to graph any link, a YES response
is typed. The calculator will then display:

WHICH LINK?

The operator now types the link number of the link to be graphed. The calculator then plots all data from the first file for that link. Before loading the next file's data, the calculator will display:

ANOTHER PLOT?

If the operator desires to plot another link or node, the response YES must by typed. This will cause the above process to be repeated until a NO is typed in response to the above prompt. At this point, the operator should check to see if the selected plots are compatible. The calculator will then load in the remaining files one at a time and plot the selected state and path values.

Of course, if a NO is typed in response to the LINK prompt, the calculator will display:

WHICH NODE?

As for the link case, the operator must now enter the node number of the state to be plotted. The calculator will then display ANOTHER PLOT and the process may be repeated. When all the desired paths and states have been graphed, the calculator links in the third plot subroutine, the label subroutine.

Label Subroutine

When the label subroutine has been linked into the calculator memory, the calculator displays:
UP, DOWN, OR CENTER?

This prompt is for positioning the plot label for each value graphed. Depending on the slope of the graph the operator may desire the label to be one space above or below the last data point calculated. The preference of the operator is entered on the keyboard, and the calculator positions the pen accordingly. The calculator then displays:

LABEL?

The operator then enters a short label for the plot, the best label being the path or state number, preceeded by a P or S. If no label is wanted, the operator merely types a space and presses the EXECUTE key. The calculator then prints the label and the process is repeated. The plots are accessed in increasing numerical order and path plots are accessed before states. Therefore, if P1, P5, S2, and S10 were being plotted, they would be accessed in the order stated.

When all the plots have been labeled, the calculator then displays:

WANT A TITLE?

The operator now enters the coordinates of the point at which a title is desired, using the graph axes as a guide. The calculator then moves the pen to the given coordinates and displays:

THIS OK?
If the operator is satisfied with the location, a YES is typed. If the operator is dissatisfied, then a NO response is typed. If the response is NO, the calculator redispays INPUT STARTING POINT and loops until the operator accepts a location for the title. When a YES response is made, the calculator displays: 

TITLE?

The operator then must type in the first line of the title. The calculator will letter the title on the graph and then display: 

ANOTHER LINE?

If the operator desires another line for the title, a YES response is typed. The calculator will position the pen at the same x-axis location, and one line lower than the last line. The calculator will then request the next line of the title. This process may be repeated as often as desired. When the last line has been lettered by the plotter, the operator must type a NO to the ANOTHER LINE prompt. The calculator will then display: 

NEW SHEET OF PAPER?

If the operator wishes to make another graph, a YES response is typed. The calculator will then link in the axis subroutine and repeat the graphing subroutines. If a NO response is entered, the calculator will display: 

FINISHED?

If the operator wants to make another simulation run, then
a NO is typed and the calculator will start another pass through the entire program. If a YES is typed, the calculator will save the model parameters on the data tape and stop execution.
Section 8 - Error Messages

The Hewlett Packard 9830A programmable calculator has a set of error codes. These codes can be found on a pull-out plastic board under the calculator keyboard. The same list of error codes can be found on page E2 in the calculator handbook.

Errors in entry by the operator generally fall into one of two categories:

1. Spelling or misentry errors
2. Logical errors

To correct misentry errors, GINS performs a check on each entered response that is of a predetermined nature (for example, yes-no answers). If the answer is not one of the possible answers, the question is asked again. The program will not continue until a proper response is made.

To check logical errors, the program checks entries for reasonableness. When entering the number of nodes and links in the model, the program checks whether the maximum number allowed (100) is exceeded. If the maximum is exceeded, the calculator displays:

TOO MANY - HOW MANY NODES?

Again, the program will wait until an acceptable number is entered. During the execution of the processor subroutine, the calculator may print:

ERROR-NO SUCH STATE MODE (2)
CHECK MODE I

where I is the state number. This error message occurs when a mode 2 constraint is encountered for a state (see pp. 64-65, Section 3 for mode descriptions). To correct the error, reassign a mode of 1 or 3 to the state.
GINS Top Level Flow Chart

Start

Input: Rerun
   Edit

Input Model

Edit Model

Process

Report?

Yes

Report Generator

No

Plot?

Yes

Plot Data

Finished

No

Yes

END
GINS File Level Flow Chart

Start

Input

Mode?

Edit

Mode?

Add

Input 1
Mark Data Files

Input 2
Load Data Arrays

Add 1
Mark Additional Data Files As Needed

Add 2
Load New Data

More Editing?

Yes

No

Tabular Data?

Yes

Tabular Data

Processor

Report?

Yes

Report Generator

More?

Yes

No

Plot?

Yes

Plot 1 Draw Axes

Plot 2 Plot Data

Plot 3 Label Graph

Finished?

Yes

No

A

END
Input 1

Start

Input No. of Nodes

Input No. of Links

Input No. of Time Increments

Mark Parameter Files (1st Tape)

Mark Data Files (1st Tape)

More Tapes?

No

Yes

Mark Data Files

Initialize P-array to $\pi$

END
Input 2

Start

Bulk Input?

Yes

Input Link Parameter Value

Input Individual Link Parameter Values

Yes

More Link Parameters?

No

Bulk Input?

Yes

Input Node Parameter Value

Input Individual Node Parameter Values

Yes

More Node Parameters?

No

Bulk Input?

Yes

Input Constraint Mode

Input Individual Constraint Nodes

No

Input Network Structure

Input Flow Equations

Save Parameters & Structure

END
Add 1

Start

Input No. of New Links, Nodes

Input No. of Time Intervals

More Tapes?

Yes

Mark New Tape & Set Flags & Counters

No

Update Counters & Flags In Previous Tapes

END
Tabular Data

Start

Insert Data Tape

Input Tabular Data for Given Link

Another Link?

Yes

No

Save on Tape File

Another File?

Yes

No

Another Tape?

Yes

No

Insert First Tape

END
Processor

Start

Load First Link File

Determine Maximum Number of Equation Iterations

Load Link File

Increment Time & Array Counters

Iterate Through Equations

Check for Link Constraint Violations

Any?

No

Yes

Check Mode & Execute

Calculate Node Values

Check for Node Constraint Violations
A

Any?

Yes

Check Mode & Execute

Save Link & Node Values

No

B

Another File?

Yes

Another Tape?

No

No

Yes

Insert Next Tape

END
Report Generator

start

Print Title

Flag links for Output

Flag Nodes for Output

Print Column Headers*

Insert Data Tape

Load Node & Link Files

Search Link Flag Vector For First Four Flaged Links

Any? Yes

Print First Four Flaged Links*

No

Search Node Flag Vector For First Four Flaged Nodes

A B
Print First Four Flaged Nodes

End of File?

End of Tape?

End of Run?

Readjust Vector Read Limits

Vector Remain?

END

*Page check made at each print statement
Plot 1

Start

Insert First Tape

Select Time Units

Determine Y-axis Range

Scale Plot

Draw & Label X-axis

Draw & Label Y-axis

END
Plot 2

Start

Load 1st Data Files For Links and Nodes

Plot Link?

Yes

Flag Link Given

No

Flag Node Given

Plot Data Point

End of File?

Yes

Another Plot?

No

C

Load Next File

A
Plot 3

start

Position Pen at End Point of Plot

Label Plot

Yes

Another Plot?

No

No

Want Title?

Yes

Input Coordinates of Title Starting Point

Confirm

Input Title

Label Graph

END
Main Program

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>DIM T50 = 160, T50 = 6, 100; V50 = 100, 100</td>
<td>Define dimension data arrays</td>
</tr>
<tr>
<td>20</td>
<td>DIM I51 = 1, I51 = 6, I51 = 100, 100; I50 = 1, 100</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>LISP &quot;INPUT&quot;</td>
<td>Prompt and select input mode, set appropriate flags</td>
</tr>
<tr>
<td>40</td>
<td>INPUT A;</td>
<td>Transfer control to input mode</td>
</tr>
<tr>
<td>50</td>
<td>IF A#&quot;NO&quot; THEN 104</td>
<td>Verify insertion of first data tape</td>
</tr>
<tr>
<td>60</td>
<td>IF A=&quot;NO&quot; THEN 30</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Z10 = Z11 = 0</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>LISP &quot;INPUT&quot;</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>INPUT A;</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>IF A=&quot;GO&quot; THEN 90</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>LOAD DATA #3 = 4</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>IF Z11 = 1 THEN 90</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>GOSUB 90</td>
<td>Prompt for first run routine</td>
</tr>
<tr>
<td>150</td>
<td>Z10 = 2</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>IF A#&quot;YES&quot; THEN 260</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>IF A#&quot;NO&quot; THEN 340</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>GOSUB 480</td>
<td>Redimension and load constraint and initial condition data</td>
</tr>
<tr>
<td>190</td>
<td>LOAD DATA #5 = 1, 7</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>LOAD DATA #4 = 2, 7</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>LISP &quot;REUSE OR EDIT&quot;</td>
<td>Prompt for rerun or edit routines</td>
</tr>
<tr>
<td>220</td>
<td>INPUT A;</td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>IF A=&quot;EDIT&quot; THEN 260</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>IF A=&quot;REUSE&quot; THEN 310</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>Z10 = 1</td>
<td>Set run flag and branch to processor, link statement</td>
</tr>
<tr>
<td>250</td>
<td>GOSUB 210</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>LISP Z10 = 0</td>
<td>Link in edit routine</td>
</tr>
<tr>
<td>260</td>
<td>STORE DATA #5 = 1, 2</td>
<td>First run routine</td>
</tr>
<tr>
<td>265</td>
<td>LOAD KEY #5 = 4</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>REWIND #5</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td>IF STATUS#5 THEN 270</td>
<td></td>
</tr>
<tr>
<td>280</td>
<td>GOSUB 430</td>
<td></td>
</tr>
<tr>
<td>285</td>
<td>GOTO 210</td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>GOSUB 90</td>
<td>Prompt for tabular flow data</td>
</tr>
<tr>
<td>300</td>
<td>INPUT A;</td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>IF A#&quot;NO&quot; THEN 310</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>IF A#&quot;YES&quot; THEN 330</td>
<td></td>
</tr>
<tr>
<td>330</td>
<td>LISP Z10 = 0</td>
<td>Last tabular data routine</td>
</tr>
<tr>
<td>335</td>
<td>LOAD DATA #5 = 2, 2</td>
<td>Set tabular data flag and save flag vector</td>
</tr>
<tr>
<td>340</td>
<td>LISP Z10 = 1</td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>LISP Z10 = 2</td>
<td>Link in processor routine</td>
</tr>
<tr>
<td>355</td>
<td>LOAD DATA #5 = 1, 2</td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>LISP Z10 = 0</td>
<td>Link in report generator</td>
</tr>
<tr>
<td>365</td>
<td>IF A#&quot;NO&quot; THEN 360</td>
<td>Prompt for print routine</td>
</tr>
<tr>
<td>370</td>
<td>IF A#&quot;YES&quot; THEN 365</td>
<td></td>
</tr>
<tr>
<td>375</td>
<td>DISP Z10 = 1</td>
<td>Link in first plot routine</td>
</tr>
<tr>
<td>380</td>
<td>LISP Z10 = 0</td>
<td>Prompt for finish routine</td>
</tr>
<tr>
<td>385</td>
<td>IF A#&quot;NO&quot; THEN 380</td>
<td>(return to start if no)</td>
</tr>
<tr>
<td>390</td>
<td>IF A#&quot;YES&quot; THEN 385</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>GOSUB 90</td>
<td>Save all input data on data tape</td>
</tr>
<tr>
<td>410</td>
<td>STORE DATA #5 = 8, 2</td>
<td></td>
</tr>
<tr>
<td>420</td>
<td>STORE DATA #5 = 5, 5</td>
<td></td>
</tr>
<tr>
<td>430</td>
<td>END</td>
<td>Load all input data from data tape</td>
</tr>
<tr>
<td>440</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>460</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

Define dimension data arrays.
Input Subroutine 1

LOAD2
LISTS10, 961
510 REM
520 L=40
530 GOTO 580
540 DISP "HOW MANY NODES?"
550 INPUT Z[5]1
560 IF Z[5]1 > 100 THEN 590
570 DISP "TOO MANY!"
580 GOTO 540
590 INPUT Z[5]1
600 DISP "HOW MANY LINKS?"
610 IF Z[5]1 < 100 THEN 640
620 DISP "TOO MANY!
630 GOTO 590
640 IF Z[5]1 = 200 THEN 570
650 N=Z[5]1
660 GOTO 680
670 N=Z[5]1
680 IF X=Z THEN 710
690 Z[O]=10
700 GOTO 720
710 Z[O]=5
720 DISP "HOW MANY TIME INTERVALS?"
730 INPUT Z[7]1
750 DISP "TOO MUCH DATA"
760 INPUT Z[4]000
770 DISP "GET MORE TAPES, TYPE GO!"
780 INPUT A1
790 IF A1=GO THEN 750
800 REBIN (Z[0]-Z[5]+Z[7])
810 IF Z[7]=10 THEN 810
820 MARK 85.1-10. Mark file for Z-vector on data tape.
830 INPUT FILE DATA tape, IO number to Z.
840 FOR I=1 TO 10
850 Z[I]=I Clear file number flags.
860 NEXT I
870 STORE 10
880 STOP
890 PRINT P2:
900 MARK 85.1-10. Mark file for model data arrays vectors and link equations.
910 MARK 85.1-10. Mark tape for mark subroutine.
920 MARK 85.1-10. Mark tape for mark subroutine.
930 FOR J=1 TO Z[5]1
940 FOR K=1 TO Z[7]1
950 IF J=K THEN JAP
960 NEXT K
970 NEXT J
980 NEXT I
990 NEXT I
100 NEXT I

Input a check number of nodes

Input a check number of links

Determine number of time intervals per file based on number of nodes and links

Input number of time intervals for run

Notify operator of need for more data tapes

Check for single tape overflow

Mark files for model data arrays vectors and link equations

Initialize P array to M for processor
Input Subroutine 1 (cont.)

320 STORE DATA# 2, 6, 10
970 FOR I = 1 TO 3
990 IF I = 1 THEN 1000
1000 NEXT I
1010 NEXT I
1020 LINK * 510, 810
1130 END

1040 MARK 15:50+1500 Mark output data file for first data tape.
1050 IF * 2 = INT(277.1) THEN 1300
1060 X = 13
1070 Z(3) =
1080 Z(4) =
1090 STORE DATA# 2, 1, 2
1100 DISP "INSERT NEXT TAPE--TYPE Go"
1110 INPUT #2
1120 IF #2 = "Go" THEN 1180
1130 PMITH #2
1140 IF SFT82# THEN 1130 Mark counter flag vector file
1150 MARK 15:1+10
1160 MARK 15:1+1.0 Increment tape counter
1170 DISK 16:1+14001 Must list files
1180 MARK 15:1+1.0 Mark tape files
1190 IF #2 = "End" THEN 1210 Need more tape?
1200 INPUT #2 Set file counter
1210 GOTO 1270
1220 IF #2 = "End" THEN 1360 Last file completely filled?
1230 Z(2) = 11.1
1240 GOTO 1260
1250 GOTO 1260
1260 END
1270 Z(3) = 11.1 First data tape
1280 Z(4) = 11.1 Set first data tape cumulative file counter
1290 STORE DATA# 2, 6, 10 Set cumulative file counter for remaining data tapes
1300 DISP "INSERT FIRST TAPE--TYPE Go"
1310 INPUT #2 Insert + verify first data tape
1320 IF #2 = "Go" THEN 1280
1330 LOAD DATA# 9, 7.5
1340 IF SFT82# THEN 1360
1350 GOTO 1400
1360 GOTO 1400
1370 GOTO 1370 Calculate number of time increments
1380 GOTO 1370
1390 GOTO 1370 Calculate number of files
1400 RETURN
Input Subroutine 2

210 FIND #51.
220 FOR J=1 TO 3
Start loop to input link initial conditions
230 IF TRUE THEN 250 Link initial condition input if completed
240 DISP "LINK INITIAL CONDITION ALL?" Input link initial condition as a whole?
250 INPUT SI
260 IF "YES" THEN 270 No input separately, Yes - input all at once.
270 IF "NO THEN 230 Improper response, re-prompt
280 IF TRUE THEN 290 Input upper constraint all at once?
290 DISP "UPPER LINK CONSTRAINT ALL?" Input link upper constraint all at once?
300 INPUT UA
310 GOTO 500
320 IF TRUE THEN 330 Skip lower constraint input if completed
330 DISP "LOWER LINK CONSTRAINT ALL?" Input link lower constraint all at once?
340 INPUT LA
350 GOTO 500
360 DISP "VALUE?"
370 INPUT V
380 FOR J=1 TO 61 Set all entries to input value.
390 T=J
400 NEXT J
410 GOTO 220
420 FOR J=1 TO 61
Set each entry to separate value.
430 J=J+1
440 NEXT J
450 NEXT J
End of link loop
460 STORE WITH NAME! Save input link data.
470 FOR J=1 TO 3 Loop node input for initial constraints and constraints
480 IF TRUE THEN 490 Input initial condition input if completed
490 DISP "NODE INITIAL CONDITION ALL?" Input initial condition all the same?
500 INPUT NA
510 IF "YES" THEN 520 Improper response, re-prompt
520 IF TRUE THEN 530 Skip upper node constraint input if completed
530 DISP "UPPER NODE CONSTRAINT ALL?" Input node upper constraint in bulk?
540 INPUT NA
550 GOTO 420
560 IF TRUE THEN 570 Improper response, re-prompt
570 DISP "LOWER NODE CONSTRAINT ALL?" Input node lower constraint in bulk?
580 INPUT LA
590 GOTO 420
600 DISP "VALUE?"
610 INPUT V
620 FOR J=1 TO 53 Bulk node input
630 UE:J=V
640 NEXT J
650
Input Subroutine 2 (cont.)

```
220 GOTO 1040
1000 FOR J=L2+1 TO H51
1010 DISP "NODE":J;
1020 INPUT UC1:J;
1030 NEXT 1
1040 INPUT J
1050 NEXT 1
End of code loop
1060 STORE UC1;J;
Show input node data
1070 DISP "CONSTRAINT MODE ALL?"
1080 INPUT M;
1090 IF M="NO" THEN 1120
1100 IF M="YES" THEN 1060
1110 INPUT J
1120 IF J(H5) THEN 1220
1200 IF J(H6) THEN 1060
End loop for inputing constraint mode (no. of nodes > no. of links)
1300 DISP "CONSTRAINT MODE ALL?
1310 INPUT M"
1320 IF M="NO" THEN 1220
1330 IF M="YES" THEN 1060
Separate constraint mode input routine
1340 GOTO 1280
1250 FOR J=1 TO H6
1260 IF J=J(H5) THEN 1220
1270 NEXT 1
Begin loop for inputing constraint mode (no. of links > no. of nodes)
1280 PRINT "Input network configuration data.
1290 PRINT "Print network definition (links)
1300 PRINT "Print network definition (nodes)
1310 PRINT "Prompt for flow equations.
1330 PRINT "Rewind data tape & load constraint vector
1350 GOTO 1410
Go to regular flow prompt in link routine.
```

Edit Change Subroutine

510 FOR J=1 TO ZC61+1
515 DISP "CHANGE;ADB"
520 INPUT #5
530 IF #5=1 THEN 570
540 IF #5=2 THEN 580
550 GOTO 515

560 IF #5=0 THEN 590
560 INPUT #5
560 IF #5="NO" THEN 700
560 IF #5="YES" THEN 630

570 M=1
580 H=ZC61
590 GOTO 730
600 DISP "WHICH_LINK"
610 INPUT J
620 M=J
630 DISP "STRUCTURE"
640 INPUT #5
650 IF #5="NC" THEN 790
650 IF #5="YES" THEN 770

660 IF J=H TO H2
670 DISP "LINK":J:"FROM TO"!!
680 INPUT CI;CJ;CI+1;CJ+1
690 NEXT J
700 DISP "INITIAL_CONDITIONS"
710 INPUT #5
720 IF #5="NO" THEN 790
720 IF #5="YES" THEN 770
730 DISP "VALUE"
740 INPUT X
750 FOR J=1 TO H2
760 T1;J+1
770 NEXT J

780 DISP "CONSTRAINT"
790 INPUT #5
800 IF #5="NO" THEN 790
810 IF #5="YES" THEN 770
820 DISP "UPPER LIMIT";
830 INPUT X
840 DISP "LOWER LIMIT";
850 INPUT Y
860 FOR J=1 TO H2
870 T1;J+1
880 NEXT J

890 DISP "CONSTRAINT_MODE";
900 INPUT #5
910 IF #5="NO" THEN 790
920 IF #5="YES" THEN 770
930 FOR J=1 TO H2
940 DISP TH6E MODE";
950 INPUT Y
960 NEXT J
970 RCJ=RCJ1=INT(RCJ1/100)*100+100*Y
980 NEXT J
Edit Change Subroutine (cont.)

100 IF HIGH THEN 1140 Edit all mode?
110 DISP "Another link parameter?" Branch to edit or save
120 IF A$="YES" THEN 790
130 IF A$="NO" THEN 1160
140 STORE DATA 72.1:1 Save edited link parameters if finished equation routine
150 GDOS 1200 Goto Flow equation routine
160 END

170 DISP "Edit all mode (nodes) Select proper mode & check for entry errors
180 INPUT J
190 IF A$="NO" THEN 1240
200 IF A$="YES" THEN 1170
210 H=1
220 H=H+1
230 GOTO 1290
240 DISP "Which node?" Set loop limits for individual edit mode
250 INPUT J
260 H=H+1
270 DISP "Initial condition?" Prompt - edit node initial condition?
280 INPUT P
290 IF A$="NO" THEN 1320
300 IF A$="YES" THEN 1370
310 DISP "Value?" Branch & check for entry errors
320 INPUT X
330 FOR J=1 TO M2 Edit initial condition to new value (node)
340 U=J+1
350 NEXT J
360 DISP "Constraint?" Prompt - edit node constraints?
370 INPUT R
380 IF A$="NO" THEN 1480
390 IF A$="YES" THEN 1320
400 DISP "Upper limit?" Prompt - input upper constraint
410 INPUT V
420 DISP "Lower limit?" Prompt - input lower constraint
430 INPUT Y
440 FOR J=1 TO M2 Edit constraints to new values
450 U=J+1
460 NEXT J
470 DISP "Constraint mode?" Prompt - edit node constraint mode
480 INPUT R
490 IF A$="NO" THEN 1570
500 IF A$="YES" THEN 1320
510 DISP "New mode?" Prompt - input constraint mode
520 INPUT V
530 FOR J=1 TO M2 Change constraint mode values
540 U=J+1
550 NEXT J
560 IF HIGH THEN 1290 Edit all mode?
570 DISP "Another link parameter?" Prompt - edit another node parameter?
580 INPUT N
590 IF A$="YES" THEN 1240 Branch to edit or save
600 IF A$="NO" THEN 1320
610 STORE DATA 72.1 Save edited node parameters
620 GDOS 1200 Goto Flow equation entry
630 DISP "Enter Flow Coefficients" Flow equation entry
640 PRINT "See Operating Manual (Cont1590)"
650 DIS "See Operating Manual (Cont1590)" Stop
660 STOP
670 DISP "More editing?" Prompt - need to do more editing?
680 INPUT N
690 IF A$="NO" THEN 500 Branch & check for entry errors
700 IF A$="YES" THEN 1150
710 GDOS 1200 Start edit routine over
720 END
Edit Add Subroutine 1

580 L1=Z16
590 L2=Z16
600 DISP "TOO MANY NEW LINKS?" Prompt number of links being added
610 INPUT P ; Input number of increment link count
620 Z16=L1+1
630 IF Z16 = 190 THEN 300
640 DISP "TOO MANY-1" ; Check for too many links
650 GOTO 610
660 DISP "TOO MANY NEW NODES?" ; Prompt number of nodes being added
670 INPUT P ; Input number of increment node count
680 Z16=L1+1
690 IF Z16 = 190 THEN 720
700 DISP "TOO MANY-1" ; Check for too many nodes
710 GOTO 660
720 DISP "TOO MANY TIME INTERVALS?" ; Prompt number of time increments
730 INPUT P ; Calculate number of files required
740 Y=Y+1; ; Z16=1
750 WAIT
760 IF Y=1 THEN 700
770 Y=Y+1; ; Z16=2
780 IF Y=4 THEN 700
790 IF A=1 THEN 340 ; Skip tape request if run shorter than last run
800 IF A=2 THEN 350 ; Skip also if new run will hit on existing tape(s)
810 DISP "TOO MUCH DATA" ; Request more tape
820 WAIT 500
830 WAIT 500
840 IF Y=1 THEN 15 ; Z16=1 THEN 670
850 L3=Z16
860 GOTO 880
870 L3=Z16
880 IF L3<90 THEN 910
890 168;B18;0
900 GOTO 880
910 Z16=12
920 G0040 (L3+L1)=Z16 \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) ; Read in data array 930 G0040 (L3+L1)=Z16 \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) \( L1=Z16 \) ; Read in second edit add subroutine 950 END
960 DISP "INSERT LAST DAT TAPE TYPE GO" ; Prompt - insert last data tape
970 INPUT F# ; Verify proper tape insertion
980 IF F)#="GO" THEN 560
990 LOAD DATA 950;0;2
1000 IF T1<Z16 THEN 200
1010 IF T1<Z16 THEN 1340 ; Skip new tape marking if new run shorter than old run
1020 IF T1<Z16 THEN 1330 ; Skip first tape marking if last tape is not type 2
1030 IF T1<Z16 THEN 1330
1040 300=300+1
1050 GOTO 110
1060 Z2=Z16
1070 GOTO 110
1080 IF X # = Z16+Z16+Z16 THEN 1540 ; Branch if run fits current tapes
1090 B=Z16
1100 DI:3=15
1110 Z14=Z16+1=0
1120 GOTO 1170
1130 Z4=Z16+1=13
1140 Z4(Z16+1)=P
1150 Z4=Z16
1160 GOTO 1520
1170 Z14=Z16
Edit Add Subroutine 1 (cont.)

1720 LET Z12=0  Reset last type flag
1730 LET Z11=110  Update node line counters
1740 LET Z10=-11  Update tape counter
1750 STORE DATA 15,0,2  Save line counters
1760 FNIND #5
1770 DISP "INSERT NEXT TAPE--TYPE GO"  
1780 INPUT #5
1790 IF #5="GO" THEN 1230
1800 IF STAT1#5 THEN 1270
1810 MARK #5-110 T Mark line for counter file
1820 IF Z10=2 THEN 1440  Increment tape counter
1830 IF STAT1#5 THEN 1500
1840 LOAD DATA 15,0,2  Load data file
1850 IF C Command THEN 1420
1860 X=INT((256-X10)/8)-1  
1870 IF X=5 THEN 1630  Is next tape last tape?
1880 GOTO 1620  
1890 X=INT(X/15)+1  
1900 IF X<>0 THEN 1610  Calculate number of used files on last tape.
1910 IF X<>0 THEN 1620  
1920 IF Z12<>1 THEN 1410  Set last tape flag
1930 IF Z11=1 THEN 1410
1940 Z11=Z11+1  Set line counters
1950 GOTO 1420
1960 Z10=Z10+15+(Z11-2)*15+Z13  
1970 GOTO 1500
1980 IF X<>0 THEN 1460  Need last data tape?
1990 GOSUB 1520  
2000 IF Z10=115 THEN 1420  Prompt for insertion of next to last data tape-repet.
2010 GOTO 1420
2020 Z10=115
2030 IF Z12=1 THEN 1430  Set last tape flag
2040 U1=1+Z13  
2050 Z10=Z10+1  
2060 GOTO 1500
2070 GOTO 1500
2080 GOSUB 1520
2090 Z10=Z10+18  
2100 Z11=Z11+1  
2110 STORE DATA #5-110  Save line counters
2120 IF Z10=111 THEN 1510  If not last tape, repeat updating process
2130 GOTO 1500  
2140 Z10=INT(X/15)+1  
2150 GOTO 1500  
2160 X=INT(X/15)+1  
2170 RETURN
2180 END
2190 DISP "INSERT TAPE--TYPE GO"  
2200 INPUT #5  
2210 IF #5="GO" THEN 1520
2220 FNIND #5
2230 IF STAT1#5 THEN 1660
2240 Z10=Z11+1
2250 LOAD DATA #5-110
2260 IF Z10=111 THEN 1520
2270 RETURN
2280 END
Edit Add Subroutine 2

580 FOR I=1 TO 3
590 DISP "LINK INITIAL CONDITION ALL?" Prompt - same value for all new link initial cond.? 
600 IF A# = "NO" THEN 590 Branch based on response
610 INPUT A#
620 IF A# = "YES" THEN 700 keyboard entry error - branch to appropriate prompt
630 GOTO 1 OR 168 OR 1790 Input response * jump to branch subroutine
640 INPUT A# Input response * jump to branch subroutine
650 GOTO 520
660 IF A# = "YES" Then 680
670 DISP "UPPER LINK CONSTRAINT ALL?" Prompt - same value, all new upper link constraints?
680 INPUT A#
690 GOTO 520 Input response * jump to branch subroutine
700 IF A# = "YES" Then 680
710 DISP "LOWER LINK CONSTRAINT ALL?" Prompt - same value, all new lower link constraints?
720 INPUT A#
730 GOTO 520 Input response * jump to branch subroutine
740 IF A# = "YES" Then 680
750 DISP "VALUE?" Prompt - new model parameter value
760 FOR J=1 TO 2
770 INPUT Y
780 NEXT J
790 GOTO 520 Input new link parameter value
800 FOR J=2+1 TO 261 Input new link parameter value
810 NEXT J
820 GOTO 520
830 IF A# = "YES" Then 680
840 DISP "NODE INITIAL CONDITION ALL?" Prompt - same value for all new node initial cond.? 
850 IF A# = "NO" THEN 860 Branch based on response
860 IF A# = "YES" THEN 900 keyboard entry error - branch to appropriate prompt
870 INPUT A#
880 IF A# = "YES" THEN 920 Input response * jump to branch subroutine
890 GOTO 860
900 IF A# = "YES" Then 920
910 DISP "UPPER NODE CONSTRAINT ALL?" Prompt - new upper node constraint same all nodes?
920 INPUT A#
930 IF A# = "NO" THEN 950 Input response * jump to branch subroutine
940 GOTO 860
950 IF A# = "YES" Then 950
960 DISP "LOWER NODE CONSTRAINT ALL?" Prompt - new lower node constraint same all nodes?
970 INPUT A#
980 GOTO 860 Input response * jump to branch subroutine
990 DISP "VALUE?" Prompt - new model parameter value
1000 FOR J=1 TO 511 Input new node parameter value
1010 UI J=Y
1020 NEXT J
1030 GOTO 860
1040 FOR J=1+1 TO 261 Input new node parameter value
1050 NEXT J
1060 GOTO 860
1070 FOR J=1+1 TO 261 Input new node parameter value
1080 NEXT J
1090 NEXT J
1090 NEXT 1
Edit Add Subroutine 2 (cont.)

100 DISP "CONSTRAINT MODE: ALL?" \ Prompt - same constraint mode for all new links + nodes?
120 IF RH="NO" THEN 1250 \ Branch - test for entry errors
1200 INPUT RH
1300 IF RH="YES" THEN 1100
1400 DISP "VALUE?" \ Prompt - input common constraint mode.
1500 INPUT H
1600 R1=INT(ABS(R1)+100) \ Link constraint mode.
1700 R2=INT(ABS(R2)+100) \ Node constraint mode.
1800 FOR I=1 TO 25 \ Add new node constraint mode.
1900 R1=INT(ABS(R1)+100)+100+Y
2000 NEXT I
2100 FOR I=12+1 TO 25 \ Add new link constraint mode.
2200 R1=INT(R11/100)+100+X
2300 NEXT I
2400 GOTO 1350
2500 FOR I=1+1 TO 25 \ Add new individual node constraint mode.
2600 DISP "NODE CONSTRAINT MODE:III"
2700 INPUT H
2800 R1=INT(R11/100)+100+ABS4
2900 NEXT I
3000 FOR I=1+1 TO 25 \ Add new individual link constraint mode.
3100 DISP "LINK CONSTRAINT MODE:III"
3200 INPUT H
3300 R1=INT(R11/100)+100+ABS4
3400 NEXT I
3500 FOR I=12+1 TO 25 \ Input additional network structure
3600 DISP "LINK:III" \ Print additional network structure.
3700 INPUT C[I]=C[I][2]+A
3800 PRINT "LINK:III" --- "11"
3900 NEXT I
4000 PRINT ""
4100 PRINT ""
4200 FOR I=1+1 TO 25 \ Print node descriptions
4300 DISP "NODE:III" REPEATEDS:1
4400 INPUT H
4500 PRINT "NODE:III" --- "11"
4600 NEXT I
4700 STORE DATA #*;1;1 \ Save network parameters on data tape.
4800 EDIT FLOW EQUATIONS
4900 PRINT "ENTER FLOW EQUATIONS"
5000 INPUT F
5100 PRINT "SEE OPERATING MANUAL: CONTINUE?"
5200 DISP "SEE OPERATING MANUAL: CONTINUE?"
5300 STOP
5400 DISP "FINISHED?" \ Prompt - Finished with editing function?
5500 INPUT RH
5600 IF RH="NO" THEN 510 \ Branch - check for entry errors
5700 IF RH="YES" THEN 1540
5800 REMING 4%
5900 GOTO 360
6000 END
Tabular Input Subroutine

510 K=1 Set type counter
516 Z=1 Set table counter Flag
520 DISP "[INSERT TAPE-TYPE GO]" Prompt insertion of data type
530 INPUT #
540 IF #=#"GO" THEN 520 Prompt insertion of type
550 REWIND #5 Verify proper type number
560 LOAD DATA $4+3 Prompt insertion of data type
570 IF $=1 THEN 520
580 FOR J=1 TO [3] Begin the loop
590 V= [input increment] Prompt input final time increment
600 IF V=1 THEN 510 Check for final time > total run time
610 DISP "[INPUT J]" Prompt input link number for tabular data
620 INPUT J
630 DISP "[INPUT V]" Prompt input tabular value
640 INPUT V
650 IF T2=2+2(5)+1 OR T3=2(3)+1 THEN 740 File overflow?
660 X=1+2(3)+1 Set tabular value for specified time interval
700 FOR J=1 TO X Set tabular value for specified time interval
710 NEXT J
720 GOTO 520
750 DISP "[OVER]" File overflow - set assignment loop termination to maximum value
760 IF T2=2(3)+1 THEN 810 Check for file overflow
770 IF T3=1 THEN 810 Check for end of run
780 NEXT J End of assignment loop
790 DISP "[FINAL] Final time increment" Prompt input new final time increment
800 GOTO 520
810 DISP "[MORE LINKS]?" Prompt other links for tabular data?
820 INPUT #
830 IF #=#"NO" THEN 860
840 IF #=#"YES" THEN 810
850 GOTO 700
860 IF K=1 THEN 990 Save tabular data
870 STORE DATA $5+2 Prompt insertion of first data type
880 GOTO 900
890 STORE DATA $5+2 Prompt insertion of first
900 NEXT K End of loop
910 IF T2=2(3)+1 THEN 940 Check for last data type
920 IF K=1 THEN 440 Increment type counter
930 GOTO 550
940 DISP "[INSERT FIRST TAPE-TYPE GO]" Prompt insertion of first
950 INPUT #
960 IF #=#"GO" THEN 940 data type
970 REWIND #5 Verify proper type number
980 IF #=2 THEN 990
990 GOTO 590 Link in processor
1000 END
Tabular Report Subroutine

```
510 DISP "TEAR OFF EXCESS PAPER"
520 WAIT 7000
530 D1=0
540 Y=0
550 [I=1 TO 5] Initialize print loop counters
560 PRINT
570 GOSUB 2340
580 PRINT
590 GOSUB 2340
600 PRINT
610 GOSUB 2340
620 PRINT "PRINTING ""GINS REPORT"
630 GOSUB 2340
640 PRINT
650 GOSUB 2340
660 DISP "HEADING"
670 INPUT A#
680 IF LEN(A#)>50 THEN 660
690 PRINT TAB(40-INT(LEN(A#)/2+.5))"A#"
700 GOSUB 2340
710 PRINT
720 GOSUB 2340
730 PRINT TABB5,"SYSTEMS ANALYSIS BRANCH"
740 GOSUB 2340
750 PRINT "CODE 2243"
760 GOSUB 2340
770 PRINT
780 GOSUB 2340
790 DISP "DATE"
800 INPUT A#
810 PRINT TAB(40-INT(LEN(A#)/2+.5))"A#"
820 GOSUB 2340
830 PRINT
840 GOSUB 2340
850 PRINT
860 GOSUB 2340
870 DISP "PRINT WHICH LINKS" I
880 INPUT A#
890 IF A#="ALL" THEN 340
900 FOR J=1 TO 2000
910 IF I(J)=2.9979
920 NEXT J
930 GOTO 1000
940 IF A#="NONE" THEN 1000
950 FOR I=1 TO LEN(A#)
960 IF A#[I]="S" THEN 270
970 NEXT I
```
Tabular Report Subroutine (cont.)

200 X=POS(RA:1,*)
990 IF X=0 THEN 1020
1000 B=RA
1010 GOTO 1030
1020 IF B=RA(1,1)+1 THEN 1030
1030 IF VAL((B))=2.9979 THEN 1400
1040 RA=RA+1
1050 IF X=0 THEN 1080
1060 GOTO 990
1070 I=S, 1 TO 50
1080 DISP "PRINT TIME FOR NODES:" Prompt select nodes for output tabulation
1090 INPUT N
1100 IF N="ALL" THEN 1150
1110 FOR J=1 TO 25
1120 UT1: J=2.9979 Flag all links for output
1130 NEXT J
1140 NEXT I
1150 FOR I=1 TO 1000 Branch if response is "none"
1160 IF N="" THEN 1230
1170 IF N="(I) THEN 1090 Check for entry error
1180 NEXT I
1190 X=POS(RA:1,"
1200 IF X=0 THEN 1230
1210 B=RA
1220 GOTO 1240
1230 IF B=RA(1,1) THEN 1250
1240 U=J, VAL(B))=2.9979
1250 RA=RA+1
1260 IF X=0 THEN 1300
1270 GOTO 1230
1280 L=55
1290 S=S+1 Node print-loop start count to avoid node print-loop
300 I=L, 1 TO 200
310 IF I=25 THEN 1300 AND CYL (="5151" IF L=1) THEN 1370
320 PRINT "" Skip line, set line count parameter
325 COGSU, 25400 Prompt for data tape insertion
330 IF CI=2 AND CYL=0 THEN 1390 Finished
340 COGSU, 25400 Read header record from tape
345 D=DISK TAPE FILE TYPE, GOTO Prompt for data tape insertion
370 INPUT N
380 IF N="NO" THEN 1390 Verify proper response to tape insertion
390 REWIND #5
400 IF STRAT=5 THEN 1400
410 LOAD DATA #5,0,2
420 IF X=0 THEN 1430
430 FOR I=1 TO 25
440 IF Ci=1 THEN 1500 Check for first data tape
450 IF CI=2 THEN 1540 Skip link data loading if link printing completed
460 LOAD DATA #5,25,1,0
470 GOTO 1510
1500 GOTO 1570

Tabular Report Subroutine (cont.)

2180 GOSUB 2340  Set header counter
2190 PRINT                   Space + call line + counter + page subroutine
2200 IF I = 27 THEN 2290    Skip if all data printed
2210 PRINT                  Print time interval header
2220 IF Y:1:1 THEN 2240     If all line printed, skip to node header
2230 FOR ILL TO 255         Begin line search loop
2240 IF I:1:202,379 THEN 2250    Check for link print flag
2250 PRINT                  Print link (print header)
2260 IF 0:0=I:2:1 THEN 2270  Increment header counter
2270 IF O:O=I:2:1 THEN 2270  Maximum header count exceeded?
2280 PRINT                  Print tabular data for next header
2290 NEXT 2:1 End of line search loop

2300 GOTO 2300

3300 PRINT                   Begin node search loop
3310 IF Y:1:202,379 THEN 2320 Check for node print flag
3320 PRINT                  Print node (node) header
3330 O:O=I:1:202,379-2:1:1 Increase header counter
3340 IF O:O=I:2:1 THEN 2300 Maximum header count exceeded?
3350 PRINT                  Print tabular data for next header
3360 NEXT 2:1 End of node search loop
3370 PRINT
3380 GOSUB 2340  \ double space + call page + line count subroutine
3390 PRINT
3400 GOSUB 2340
3410 RETURN

3500 IF I:2=0 THEN 3500     Check for end of page or last link
3510 IF I:2=0 THEN 3500     Any node left to be checked?
3520 EQ:0=9-01              Calculate number of blank lines for end of last page
3530 GOTO 2440
3540 PRINT                  Increment page counter
3550 FOR I = 1 TO 6         Space to page number
3560 PRINT                  Print page number
3570 NEXT 6
3580 PRINT                  Space to end of page
3590 PRINT                  Page end subroutine
3510 RETURN
Tabular Report Subroutine (cont.)

1310 LOAD DATA #5:27F:3  Load node data
1320 GOTO 13.0
1330 IF L1:10 THEN 1350  Skip last data load if line data printing completed
1340 LOAD DATA #5:27F:4:1  Load line data
1350 GOTO 13.0

1570 FOR J=1 TO #211 Begin array row loop
1580 IF F=0:0 THEN 1590 Calculate print time interval
1590 F=(F-1+32:1+1)$2:1+1-1  Calculate print time interval
1600 PRINT F;B;T;  
1610 IF L1:10 THEN 1630 Skip to node print, if line print completed
1620 IF T1:1:1:1#2:1:1 THEN 1670 Check for print flag for links
1630 IF 02=01:1 THEN 1680 Increment column counter
1640 IF 02=01:1 THEN 1690 Maximum column count exceeded?
1700 IF I=J THEN 1720 Print link data
1710 PRINT CHM+INT(ABS(N2));  
1720 NEXT J  End of search loop
1730 PRINT Space between rows of table entries
1740 IF F=0:1 THEN 2000 Last data value?
1750 IF F=0:1 THEN 2000 Print node data
1760 IF I=J THEN 2100 Skip to end if node print completed
1770 PRINT  
1780 FOR J=1 TO #211 Begin node search loop
1790 IF L=I:1:1:1 THEN 2020 Check for print flag for nodes
1800 IF 02=01:1 THEN 2100 Increment column counter
1810 IF 02=01:1 THEN 2100 Maximum column count exceeded?
1820 PRINT  
1830 NEXT J  End of search loop
1840 PRINT  
1850 IF L=I:1:1:1 THEN 2100 Space between rows of table entries
1860 IF T=V:1:1 THEN 2000 Last data entry?
1870 IF T=V:1:1 THEN 2000 Print node data
1880 IF I=J THEN 2100  
1890 STORE 21  Full page print count subroutine
2010 GOTO 2100  
2020 NEXT I  End of array row loop
2030 NEXT I  End of file loop
2040 PRINT  
2050 IF J=1:1 THEN 2060 Increase tape counter
2060 GOTO 1390  
2070 IF 03=0 THEN 2080 More columns left to print?
2080 K=1  Reset tape counter
2090 GOTO 1390  
2070 GOTO 2100  
2100 GOTO 312  
2110 END
Axis Subroutine

Prompt: ready to plot

Set angular mode

Prompt: insert first data tape

Verify proper tape invention

Prompt: assign time units

Prompt: available time units

Select time units

Select time (x-axis) scale

Draw x-axis

Label x-axis

Prompt i input maximum y-axis data point

Prompt i input minimum y-axis data point

Calculate y-axis scale

Label y-axis
Data Plot Subroutine

510 FOR J=1 TO 20
520 TL=J
530 FOR J=3 TO 28:530 UC(J)=J
540 NEXT J
550 NEXT J
560 NXT=
570 REL=0
580 IF REL=1 THEN 620
590 LOAD DATA #5,1:590 LOAD DATA #5,2:590 LOAD DATA #5,3:
610 GOTO 640
620 LOAD DATA #5,4:620 LOAD DATA #5,5:620 LOAD DATA #5,6:
640 DISP "LINK?"
650 INPUT #:
660 IF #="NO" THEN 740
670 IF #="YES" THEN 640
680 DISP "WHICH LINK?"
690 INPUT #:
700 J=VAL(#):
710 TL=J
720 NEXT J
730 GOTO 760
740 DISP "GET NODE?"
750 INPUT #:
760 J=VAL(#):
770 NEXT J
780 GOTO 999
790 FOR J=1 TO 28
800 LOAD DATA #J,1:800 LOAD DATA #J,2:800 LOAD DATA #J,3:
810 IF POSDATA("NO") THEN 850 Plot lines at nodes
820 PLOT TL,J Plot data line
830 IF J=24 THEN 890 Store data point in case of file jump
840 PLOT TL,J Plot data (node)
850 UC(J)=TL,J Store data point in case of file jump
860 NEXT J End of first file plot loop
870 PLOT TL,J Plot data
880 NEXT J
890 DISP "ANOTHER PLOT?"
900 INPUT #:
910 IF #="NO" THEN 370
920 IF #="YES" THEN 280
930 NEXT J
940 NEXT J
950 GOTO 640
960 NEXT J
970 NEXT J
980 IF K=1 THEN 1050
990 NEXT J
1000 DISP "INSERT TAPE IN "-TYPE GO"
1010 INPUT #:
1020 IF #="GO" THEN 1050
1030 LOAD DATA #5,6:2
1040 IF TL=J THEN 1050
1050 Save last time value
1060 IF K=1 THEN 1050
1070 NEXT J
1080 NEXT J
1090 NEXT J
Prompt - another plot on same page?
Prompt - insert data tape & carry on
Data Plot Subroutine (cont.)

1050 FOR F=1 TO 3
1050 IF ZIF.5=0 THEN 1230
1070 LOAD DATA #5,F,1
1090 GOTO 1130
1090 LOAD DATA #5,F,1
1100 LOAD DATA #5,F,1
1120 IF ZIF.5=1 THEN 1220
1140 IF ZIF.5=0 THEN 1230
1150 FLOT 0,1,1,1 Plot last value
1170 IF ZIF.5=0 THEN 1230
1190 IF ZIF.5=1 THEN 1210
1210 LOAD DATA #5,F,1
1230 IF ZIF.5=0 THEN 1230
1250 LOAD DATA #5,F,1
1270 LOAD DATA #5,F,1
1290 END
Label Subroutine

LOAD LIST

510 LABEL "LOADING" Reorient axis
520 FOR I=1 TO 150
530 IF T(I,1).EQ.2 THEN GOTO 460 Check for plot flags (links)
540 PLOT PLOT(0,0,0,1,1) Plot last data point
550 GOTO 200 Call plot label subroutine
560 NEXT I
570 FOR J=1 TO 150
580 IF U(J,1).EQ.2 THEN GOTO 450 Check for plot flags (nodes)
590 PLOT PLOT(-1,1,1,0) Position pen for label
600 GOTO 200 Call plot label subroutine
610 NEXT J
620 DISP "WHAT TITLE?" Prompt request for title: check for entry error
630 INPUT AF
640 IF AF="NO" THEN 810
650 IF AF="YES" THEN 420
660 DISP "INPUT STARTING POINT" Prompt request title location
670 INPUT X,Y
680 PLOT X,Y Position pen at given title location
690 DISP "THIS EX" Prompt verify title position acceptability
700 INPUT AF
710 IF AF="NO" THEN 460
720 IF AF="YES" THEN 480
730 ON (Set line counter
740 DISP "TITLE" Prompt - type in graph title
750 INPUT AF
760 LABEL "RETURN TO LABEL" Print title with plotter
770 DISP "ANOTHER LINE?" Prompt - request for more title
780 INPUT AF
790 IF AF="NO" THEN 810
800 IF AF="YES" THEN 210
810 PLOT X,Y Position pen to start pen for next line
820 GOTO 210 Increment line counter
830 GOTO 290
840 DISP "NEW SHEET OF PAPER?" Prompt - More paper for another graph?
850 INPUT AF
860 IF AF="NO" THEN 870
870 IF AF="YES" THEN 490
880 LINE 16910,1 Line in Plot Axis routine
890 END
900 GOTO 330 Go to Finish routine in Link routine
910 END
920 DISP "OFF DOWN FROM CENTER" Prompt - select position of plot label
930 INPUT AF
940 IF AF="NO" THEN 950
950 IF AF="YES" THEN 890 Branch to selected position statement
960 CPlot 0,1 Down position
970 GOTO 960
980 CPlot 0,1 Up position
990 GOTO 960
1090 CENTER position
1100 DISP "CHILL" Prompt - type plot label
1110 INPUT AF
1120 LABEL "LOADING" Print plot label with plotter
1130 RETURN
1000 END