AN ABSTRACT OF THE THESIS OF

WILLIAM LLOYD STUBKJAER for the MASTER OF SCIENCE
(Name) (Degree)

Electrical and in Electronics Engineering presented on 28 NOVEMBER 1967
(Major) (Date)

Title: A LINEAR CIRCUIT ANALYSIS PROGRAM FOR A SMALL
COMPUTER WITH NO AUXILIARY MEMORY

Abstract approved: Redacted for privacy

Donald Amort

A set of circuit analysis programs designed for the IBM 1620 computer (having 40,000 digits of memory) is described and its operation is explained. The set consists of d-c, a-c, and transient analysis programs.

The d-c and a-c analysis programs provide the branch current and node-to-reference potential for linear circuits. They have input/output formats similar to those for ECAP (Electronic Circuit Analysis Program) but are more limited in scope. They provide either for batch processing or on-line modification of circuits.

The transient analysis program provides the branch currents and node-to-reference voltages for linear passive circuits. It is based on a state variable analysis and, because of its length, can analyze only one circuit with each loading.

Solutions from the a-c and d-c programs when rounded to five
digits agree with the solutions from ECAP. The transient test solutions agree to three digits; however, since ECAP can only approximate the reactive elements, the transient analysis may be more accurate than initially indicated.
A Linear Circuit Analysis Program for a Small Computer with no Auxiliary Memory

by

William Lloyd Stubkjaer

A THESIS

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Master of Science

June 1968
APPROVED:

Redacted for privacy

Associate Professor of Electrical and Electronics Engineering
in charge of major

Redacted for privacy

Head of Department of Electrical and Electronics Engineering

Redacted for privacy

Dean of Graduate School

Date thesis is presented ________________

26 November 1967

Typed by Clover Redfern for ____________ William Lloyd Stubkjaer ____________
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>D-C ANALYSIS PROGRAM</strong></td>
<td>3</td>
</tr>
<tr>
<td>Forming Equations</td>
<td>3</td>
</tr>
<tr>
<td>Input Format</td>
<td>6</td>
</tr>
<tr>
<td>Operation</td>
<td>9</td>
</tr>
<tr>
<td>On-Line Modification</td>
<td>10</td>
</tr>
<tr>
<td><strong>A-C ANALYSIS PROGRAM</strong></td>
<td>16</td>
</tr>
<tr>
<td>Input Format</td>
<td>16</td>
</tr>
<tr>
<td>Output</td>
<td>20</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>20</td>
</tr>
<tr>
<td>On-Line Modification</td>
<td>21</td>
</tr>
<tr>
<td><strong>TRANSIENT ANALYSIS</strong></td>
<td>26</td>
</tr>
<tr>
<td>Forming the Equations</td>
<td>26</td>
</tr>
<tr>
<td>Matrix Inversion</td>
<td>35</td>
</tr>
<tr>
<td>Solution to State Equations</td>
<td>35</td>
</tr>
<tr>
<td>Round-off and Truncation Error in Solving the State Equations</td>
<td>36</td>
</tr>
<tr>
<td>Sources</td>
<td>37</td>
</tr>
<tr>
<td>Input Format</td>
<td>39</td>
</tr>
<tr>
<td>Operation</td>
<td>43</td>
</tr>
<tr>
<td><strong>CONCLUSIONS</strong></td>
<td>47</td>
</tr>
<tr>
<td><strong>BIBLIOGRAPHY</strong></td>
<td>50</td>
</tr>
<tr>
<td><strong>APPENDIX</strong></td>
<td>51</td>
</tr>
<tr>
<td>Appendix I</td>
<td>51</td>
</tr>
<tr>
<td>Appendix II</td>
<td>61</td>
</tr>
<tr>
<td>Appendix III</td>
<td>76</td>
</tr>
<tr>
<td>Appendix IV</td>
<td>116</td>
</tr>
<tr>
<td>Appendix V</td>
<td>118</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.</td>
<td>I-th circuit block showing reference directions.</td>
</tr>
<tr>
<td>2.</td>
<td>D-C analysis example.</td>
</tr>
<tr>
<td>2a.</td>
<td>D-C equivalent circuit for a single stage common emitter amplifier.</td>
</tr>
<tr>
<td>2b.</td>
<td>Input to D-C analysis program.</td>
</tr>
<tr>
<td>2c.</td>
<td>Computer output for D-C equivalent circuit shown in Figure 2a.</td>
</tr>
<tr>
<td>3.</td>
<td>A-C analysis example.</td>
</tr>
<tr>
<td>3a.</td>
<td>Phase shift oscillator.</td>
</tr>
<tr>
<td>3b.</td>
<td>Input to A-C analysis program.</td>
</tr>
<tr>
<td>3c.</td>
<td>Computer output for A-C equivalent circuit shown in Figure 3a.</td>
</tr>
<tr>
<td>4.</td>
<td>Transient analysis example.</td>
</tr>
<tr>
<td>4a.</td>
<td>Bridge-tee circuit.</td>
</tr>
<tr>
<td>4b.</td>
<td>Input to transient analysis program.</td>
</tr>
<tr>
<td>4c.</td>
<td>Computer output for circuit shown in Figure 4a.</td>
</tr>
</tbody>
</table>
A LINEAR CIRCUIT ANALYSIS PROGRAM FOR A SMALL
COMPUTER WITH NO AUXILIARY MEMORY

INTRODUCTION

The purpose of this thesis project was to develop an electrical
circuit analysis program to be run on an IBM 1620 Model I digital
computer. The program was to be based on ECAP (Electronic Cir-
cuit Analysis Program) developed by IBM. Although it was written
for the 1620, ECAP could not be used directly because this model I
lacked sufficient memory. It was decided to restrict the program to
finding the block current and node-to-reference potential for linear
circuits. In order not to exceed memory, the program was divided
into three self-contained parts, one each for d-c, a-c, and transient
analysis.

A large class of linear d-c circuits can be analyzed, using the
first of the three programs. Originally the d-c analysis was a two-
pass program, with data being transmitted by punched cards between
pass one and pass two. However, when it was discovered that the
extra memory thus made available for data could not be effectively
utilized because of time limitations, the two passes were combined.
This made it practical to include an on-line parameter modification
feature.

The a-c analysis program followed the development of the d-c
program and was also limited to linear circuits. A one-pass pro-
gram, the a-c analysis features automatic frequency incrementation
as well as the on-line parameter modification feature.

The transient analysis, which uses state equations, is restrict-
ed to linear passive circuits. Because of the size, the program was
divided into three passes; however, it was programmed so that pass
one calls pass two and pass two calls pass three. All data are trans-
mitted internally. Using state variable analysis simplified the pro-
gramming of the numerical integration necessary to compute the time
behavior of the circuit. Also it will be relatively easy to change in-
tegration routines should greater accuracy be desired. Finally, be-
cause state variable equations can be solved directly on the analog
computer, the transient analysis possibly could be implemented on a
hybrid computer to some advantage.
D-C ANALYSIS PROGRAM

Forming Equations

The d-c analysis is based on the matrix formalization of Kirchhoff's nodal equations (1, p. 77-79; 7, p. 160-1). The circuit to be analyzed is thought of as being composed of blocks of the form shown in Figure 1. The network would consist of IB of these blocks and would have (NN + 1) nodes. These blocks must be numbered, starting with one and continuing in sequence. Nodes must be numbered in sequence, starting with zero for the reference nodes. From such a specification of the circuit the following matrices can be directly formed.

\[
\frac{1}{C} = [C_k] \quad k = 1, \ldots \text{IB.} \quad \text{C is the current source matrix.}
\]

\[
Y = [1/Z_{k,k}] + [\beta_{k, \ell}/Z_{\ell, \ell}] \quad k = 1, \ldots \text{IB}; \quad \ell = 1, \ldots \text{IB}
\]

\[
Y \text{ is the branch admittance matrix.}
\]

\[
Z_{k,k} \quad \text{is a resistance.}
\]

\[
\beta_{k, \ell} \quad \text{is a current ratio such as the } \beta \text{ of a transistor.}
\]

---

1In the literature matrices are denoted by bold-faced type. In this paper matrix elements will be denoted by subscripted characters (such as \( C_j \), or \( e_i \)) or by matrices. Matrices will be represented by characters (\( C, e, AA, \) or \( F_{LG} \)) or by a matrix element enclosed by square brackets ([\( C_j \)]). The distinction between a matrix (such as \( Y \)) and a constant (such as \( \text{IB} \)) can be made from context.
Z<sub>i</sub> - A non-zero impedance.

C<sub>i</sub> - An independent current source (which may be zero).

E<sub>i</sub> - An independent voltage source (which may be zero).

(β<sub>i, ℓ</sub>)(J<sub>ℓ</sub>) - A dependent current source with an output of β<sub>k, ℓ</sub> times the current J<sub>ℓ</sub> through the impedance in the ℓ-th branch. In general, a block may contain more than one dependent current source.

J<sub>i</sub> - The element current flowing through Z<sub>i</sub>.

I<sub>i</sub> - The i-th branch current.

e<sub>i</sub> - The potential from node k to node j.

e<sub>k</sub> - The potential from node k to the reference node.

Figure 1. I-th circuit block showing reference directions.
\[ E = \begin{bmatrix} E_k \end{bmatrix} \quad k = 1, \ldots, IB \]

\[ E \] is the voltage source matrix.

\[ AA' = \begin{bmatrix} \epsilon_{k,j} \end{bmatrix} \quad k = 1, \ldots, IB; \quad j = 1, \ldots, NN \]

where

\[ \epsilon_{k,j} = 0 \] if the \( k \)-th block is not connected to the \( j \)-th node.

\[ \epsilon_{k,j} = 1 \] if the \( k \)-th block is connected to the \( j \)-th node and the reference current is away from that node.

\[ \epsilon_{k,j} = -1 \] if the \( k \)-th block is connected to the \( j \)-th node and the reference current direction is toward that node.

We can note that \( e' = (AA')(e) \) and \( (AA)(I) = 0 \). The matrix \( AA \) (often denoted in the literature by \( A \)) is called the vertex matrix. The following unknown matrices can also be formed.

\[ I = \begin{bmatrix} I_k \end{bmatrix} \quad k = 1, \ldots, IB \]

\[ I \] is the branch current matrix.

\[ e = \begin{bmatrix} e_k \end{bmatrix} \quad k = 1, \ldots, NN \]

\[ e \] is the node voltage matrix.

Using topological arguments along with Ohm's law, the following equations can be developed.

\[
(\begin{array}{cc}
AA & Y \\
\end{array}) (\begin{array}{c}
AA' \\
\end{array})(e) = (AA)(C-(Y)(E))' \tag{1}
\]

\[
I = (-C) + (Y)(E+(AA')(e)) \tag{2}
\]

Equation (1) is equivalent to a system of \( IB \) equations in the \( IB \) unknowns \( e_1', e_2', \ldots, e_{IB}' \) and can be solved by using the
Gauss-Jordan method, also known as the method of "complete elimination" (6, p. 360-2). Once the node potential vector \( \mathbf{e} \) has been found, the branch current vector \( \mathbf{I} \) is given directly from Equation (2).

**Input Format**

The input consists of a description of the blocks which make up the circuit to be analyzed. It consists only of numbers, but otherwise is similar to the input for ECAP. Free format subroutines, which were used in compiling the program, require that at least one space appear between the data fields. Other than that, however, the position of the fields on the card is not important. It is important that all fields specified are actually present. Otherwise in order to complete its input, the program will take a number from the next card and will usually halt in a check stop.

Inputs are normally made on cards. Circuits should be initially defined using cards and may subsequently be modified using the console typewriter.

The first card describes the size of the circuit, and contains three integer fields:

\[
\begin{array}{ccc}
IB & NN & IC \\
\end{array}
\]
IB is the number of blocks in the circuit and must be less than 20.

NN is the number of nodes other than the reference nodes, which are always labeled with a zero. NN must be less than 19.

IC is the number of dependent current sources. Normally this will not exceed IB.

The next IB cards describe the blocks. They must be arranged in numerical order, starting with one and continuing in sequence to IB. Each card must contain three integers followed by three floating point number fields:

\[ \begin{array}{cccccc}
I & J & K & R & D & B \\
\end{array} \]

I is the number of the block described by the card.

J is the number of the source node.

K is the number of the sink node. In block I the reference current flows from node J to node K.

R is the value of the block resistance (in ohms). R must never equal zero.

D is the value of the voltage source (in volts). If no source is present, D must be zero.

B is the value of the current source (in amps). If no source is present, B must be zero.
The final IC cards describe the dependent current sources. In general a block may contain more than one dependent current source or may contain none. Sources which depend upon the current in the block in which they are contained are also permitted. Special restrictions, discussed on page 11, apply if the circuit is to be subsequently modified by typewriter. These final IC cards contain three integer fields, followed by one floating point field:

\[
0 \ J \ K \ R
\]

0 is the character zero and must always be present.

J is the number of the controlling branch.

K is the number of the block containing the controlled source.

R is the value of $\beta$ for the source.

It is expected that a circuit will be described by $(IB+IC+1)$ cards, with the first card describing the size of the circuit. Each card must be correct as to number and type of number fields because these errors cannot be checked directly by the program and will probably cause a check stop condition, stopping computations.

Some other errors will be caught by the program, and in these cases an error message will be typed on the console typewriter. Error indications will be given if the block specification cards are not in numerical order, if the dependent source specification cards, when
required, do not follow the block specifications, if some block has zero resistance, or if the circuit is too large. Normally when an error is detected by the program, it will type an error message and then try to process the input for the next circuit.

**Operation**

The normal mode of operation is batch processing. The computer will accept data for the first circuit, compute and output the results, and then accept data for the next circuit all without operator intervention. The following instructions will usually permit batch processing:

1. Turn all sense switches off.
2. Clear the punch, using NON PROC RN OUT switch and load blank cards if needed.
3. Place d-c analysis program followed by data cards in the reader hopper.
4. Push INSTANT STOP and RESET keys on the console to set up the machine for loading the program.
5. Push the LOAD button on the reader to start the loading of the program.
6. Before the program has been loaded, push the punch START button.
7. The program will read data cards and analyze the circuits
until (1) it has read all the cards, (2) it runs out of blank cards, or (3) it finds a non-checkable error. It should be noted that in order to read the last card, it is necessary to push the reader START.

8. The output will reproduce the circuit data cards and will give the node-to-reference potential and branch current.

**On-Line Modification**

As an optional feature it is possible to observe the results of an analysis, to change some parameters in the circuit, and then to observe the results of the change. This option is selected by turning on sense switch three. The circuit to be analyzed must initially be entered by cards. The inputs and outputs are reproduced on the typewriter as well as on the cards. After the analysis is completed, the computer will accept input on the console typewriter. Sense switch one controls the input through the typewriter. The input is of the same format as that for card inputs with one exception—that dependent source specifications must contain six number fields. It is convenient to follow the value of $\beta$ with $0.0.0$.

Normally all that is desired is to change the values of parameters in the circuit. However, it is possible accidentally to affect the circuit topology through an incorrectly typed input. Thus it is very important to make sure that an input is error-free before giving an
R-S signal. Sense switch two is used to indicate an error and allow correction. If one understands the program, it is possible to change correctly circuit topology. However, this takes longer than the recommended procedure of entering new circuit configurations by using card inputs.

Some special restrictions involving controlled sources must be observed if correct results are to be obtained. A block may in general have more than one dependent current source. However, if a dependent source is to be entered by the typewriter, it must be the only one present in that block. If more than one value is entered, only the last value will affect the output of the source.

A special case occurs with a source dependent on the block in which it is contained. If the $\beta$ for this type of current source is to be altered, the controlling block data must first be reentered. Then the controlled source data can be entered. In this case any number of controlled sources can be present.

Should the resistance in a controlling block be changed, then all of the sources dependent on that block must subsequently be reentered.

With the above mentioned restrictions in mind, the following instructions will permit on-line modifications of circuits:

1. Turn sense switches one, two, and four off; sense switch three on.

2. Clear punch and load blank card.
3. Place d-c analysis program followed by data cards in reader hopper.

4. Push INSTANT STOP and RESET buttons on console.

5. Push LOAD button on reader.

6. Push punch START button.

7. After the first circuit has been analyzed, the results will be printed on the typewriter. Then SS 1 ON TO TYPE INPUT will be printed, and the machine will go into MANUAL mode.

8. If data are to be entered, turn sense switch one on and push START.
   a. TYPE INPUT will be printed.
   b. Type the new data (six numbers must be typed).
      i) If the new data are not correct, turn sense switch two on and press the R-S key on the typewriter. The program will return to 8-b to allow correction.
      ii) If the new data are correct, press the R-S key. Sense switch two should be off. The program will take the information typed and include it in the circuit matrices.
   c. Before the last line of data has been entered, turn sense switch one off, and the program will compute currents and voltage for the modified circuit and then
return to step 7.

9. If the circuit is not to be altered, sense switch one should be turned off and START pushed. The program will then request that a new circuit be entered by cards.
Figure 2. D-C analysis example.

Figure 2a. D-C equivalent circuit for a single stage common emitter amplifier (6, p. 8-11).

Figure 2b. Input to D-C analysis program.

```
6 3 1
1 0 2 2000. 20. 0.
2 0 1 6000. 20. 0.
3 0 1 1000. 0. 0.
4 1 3 350. -0.5 0.
5 3 0 500. 0. 0.
6 2 3 11.1E3 0. 0.
0 6 50.
```
Figure 2c. Computer output for D-C equivalent circuit shown in Figure 2a.
A-C ANALYSIS PROGRAM

The a-c analysis is based on the same equations as the d-c analysis except that provision has been made to handle inductors and capacitors. The analysis is made using phasor representation; hence the program must perform complex arithmetic. In particular, the Y, e, and I matrices must be made up of a real part and a complex part.

The a-c circuit must be broken down into blocks of the form shown in Figure 1. However, in this case $Z_i$ may be a resistor, an inductor, a capacitor, a parallel inductor-resistor, or a parallel capacitor-resistor. The sources are sinusoidal and may assume relative phase angles of integer multiples of $\pi$.

Since the admittance of reactive elements vary with frequency, the program will analyze circuits at several frequencies without having to reread the circuit description cards each time.

The output gives the node-to-reference potential and the branch current in phasor form--as a magnitude and a phase angle.

Input Format

The input consists of a description of the blocks which make up the circuit to be analyzed and the frequencies at which the analysis is to be made. PDQ - Free Format subroutines were used in compiling
the program.

The first card describes the size of the circuit and contains three integer fields:

\[
\begin{array}{ccc}
IB & NN & IC \\
\end{array}
\]

IB is the number of blocks in the circuit and may not exceed 14.
NN is the number of nodes, other than reference nodes, which are always labeled with zero. NN is less than 14.
IC is the number of dependent current sources.

The second card describes the range of frequencies at which the analysis is to be made. It contains three floating point number fields:

\[
\begin{array}{ccc}
WW & DW & WX \\
\end{array}
\]

WW is initial frequency (in hertz).
DW is the frequency increment.
WX is the terminating frequency.

Thus the second frequency at which analysis is made is \( DW \) times \( WW \), the third frequency is \( DW \) times the second, and so on until the frequency equals or exceeds \( WX \).

The next \( IB \) cards describe the blocks. They must be arranged in numerical order starting with one and continuing in sequence to \( IB \). Each card contains three integers followed by four
floating point number fields:

I J K R RR D B

I is the block number.

J is the source node number.

K is the sink node number.

R is the block resistance.

RR is either the block inductance (in henries) if RR is greater than zero or the block capacitance (in farads) if RR is less than zero.

D is the magnitude of the block voltage source.

B is the magnitude of the block current source.

Either of R or RR, but not both, may be zero. If the block does not contain a source, then a floating point zero must appear in the appropriate place on the card.

The final IC cards describe dependent current sources. In general a block may contain any number of dependent sources. The value of R is a real number. The controlling block may contain a reactive element. Each card must contain three integers followed by one floating point number field:

0 J K R
0 is the character zero which identified that card as specifying a dependent current source.

J is the number of the controlling branch.

K is the number of the block containing the source.

R is the value of $\beta$ for the controlled source.

Because of the free format subroutines used, care must be taken to insure that $(IB + IC + 2)$ cards are used to describe each circuit and that each card has the correct number of fields. The first and second cards must describe the circuit size and the frequency range respectively. Errors in this format cannot be checked, and their occurrence will probably halt computation with a check stop.

Other errors can be checked by the computer. Errors of this type will cause an error message to be typed and will stop the processing of the input; however, the program will read through the remaining cards until it reaches the place where the next circuit should be described. Then it will proceed to process this next circuit.

A special type of error results if $DW$ equals zero in the input specification. When this happens, $DW$ is automatically set equal to 1000, and the computation is continued. Since it was not believed that this error would be serious enough to terminate computation, a large enough value of $DW$ was chosen to insure that $(DW)(WW) \geq (WX)$. 
It should be noted that the d-c input cards cannot be used for a-c analysis and vice versa because of the difference in format statements used in the programs.

**Output**

The currents and voltages for the a-c circuit are computed as phasor quantities and represented by complex numbers. Prior to the output these numbers are converted from rectangular co-ordinates to polar co-ordinates. The complex number \( a + jb \) is converted to its polar form \( M/\theta \) by the relations

\[
\begin{align*}
\theta &= \arctan \left( \frac{b}{a} \right) \\
M &= \frac{a}{\cos(\theta)}
\end{align*}
\]

In the output \( \theta \) is given in degrees.

**Batch Processing**

As with the d-c analysis the a-c analysis is designed for batch processing. The computer will accept data for the first circuit, compute the results, and then accept data for the next circuit all without operator intervention. The processing continues until all the circuits have been analyzed or a check stop occurs. The latter usually indicates an error in the input format. In most circumstances where no errors exist, the procedure described for batch processing of d-c
circuits should also be used for a-c circuit analysis.

On-Line Modification

As an optional feature it is possible to observe the results of an analysis, to change the values of some of the elements in the circuit, and then to observe the results of the change. This option is selected by turning on sense switch three. The circuit to be analyzed must be initially entered by cards. After the analysis is completed, the computer will accept input from the console typewriter depending upon the position of sense switch one. The first input that must be typed is the frequency specification, corresponding to the second card. Next the altered block and dependent source statements are entered. They have the same format as the cards except that the dependent source specification must contain seven number fields. It is convenient to follow the value of $\beta$ with 0.000.

It is the intent of this feature that only the values of circuit elements be changed using the console typewriter. Any change in the circuit topology should be made using card input. Errors can be corrected by turning sense switch two on before typing the R-S key. This will allow the correct data to be read.

Some special restrictions must be observed in using the parameter modification option. If a circuit is to be analyzed over a range of frequencies by internal control, then blocks controlling dependent
current sources may contain only resistive elements. Should it be necessary to have reactive elements in controlling blocks, then each frequency must be treated as a separate circuit.

Irregardless of whether or not automatic frequency incrementation is used, a block may contain no more than one dependent source when that source is to be entered on the typewriter. An exception is the source controlled by its own block as mentioned in the d-c section. Finally whenever a controlling block resistance is altered, the sources dependent on that block must subsequently be re-entered.

Observing those restrictions, the procedure outlined on page 11 for the operation of the d-c analysis parameter modification feature can be followed for the a-c analysis except for the following changes:

8. If data are to be entered, turn sense switch one on and push START.

a'. TYPE INPUT (FREQ) is typed.

a''. The frequency specification must be typed using the same format as for the card input of frequency information. This data can be corrected using sense switch two as outlined below.

a. TYPE INPUT (PARAM) is typed.

b. Type the new data (seven numbers must be typed).

i) ...
Figure 3. A-C analysis example.

Figure 3a. Phase shift oscillator.
Figure 3b. Input to A-C analysis program.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000.</td>
<td>1.</td>
<td>1000.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.</td>
<td>0.</td>
<td>1.</td>
<td>0.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>10000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>39000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>600.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3</td>
<td>2700.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>0.</td>
<td>-5.</td>
<td>E-9</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0</td>
<td>1000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>5</td>
<td>0.</td>
<td>-5.</td>
<td>E-9</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td>1000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>6</td>
<td>0.</td>
<td>-3.</td>
<td>3E-9</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>0</td>
<td>600.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>0</td>
<td>1000.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0</td>
<td>0.</td>
<td>-1E-4</td>
<td>0.</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
<td>100.</td>
<td>0.</td>
<td>0.</td>
</tr>
</tbody>
</table>
A-C CIRCUIT ANALYSIS

Figure 3c. Computer output for A-C equivalent circuit shown in Figure 3a.
TRANSIENT ANALYSIS

The state variables for a system are the minimum number of parameters which completely describe the time behavior of the system. The state variable vector \( x = [x_1(t)] \) is described by the following equations:

\[
\frac{dx}{dt} = (A)(x) + (B)(u(t)) \quad (4)
\]

\[
y = (C)(x) + (D)(u(t)) \quad (5)
\]

where \( u(t) = [u_j(t)] \) are the inputs to the system and \( y = [y_k(t)] \) are the desired outputs.

For an electrical circuit it can be shown that a subset of the inductor currents and capacitor voltages form a state variable system. The outputs \( y \) are the node-to-reference potentials and the branch currents.

The transient analysis can logically be broken into two parts. It is first necessary to form the state equations from a description of the circuit. Next it is necessary to solve for the state variables and then to find the output quantities.

Forming the Equations

A few definitions are necessary before the state equations can be discussed. A tree is a collection of branches which connect all the
nodes of the circuit without forming a closed path. The branches of the tree are called tree branches, and the rest of the branches in the circuit are called links.

The circuit is thought of as consisting of blocks, also called branches, of the type shown in Figure 1, except that each block may contain only one impedance element and that dependent current sources are not allowed. Next a tree is formed from the elements such that it contains a majority of the capacitor-containing blocks and such that a majority of the inductor blocks are in the set of links. Such a tree is often called a normal tree. This tree will contain all the capacitor blocks if and only if there are no capacitor loops in the circuit. The links will contain all the inductor blocks if and only if there are no inductor cut sets. The state variables will be the tree-branch capacitor voltages and the link inductor currents.

To form a normal tree it is necessary first to choose as many as possible of the capacitor blocks which do not form a closed path. If the tree is not complete, then the resistor blocks should be added. If necessary the tree is completed with inductor blocks. The remaining blocks can be classified as either capacitor links, resistor links, or inductor links. The circuit blocks must be numbered, starting with the capacitor links first, the resistor links second, and next the inductor links, the capacitor tree branches, resistor tree branches, and finally the inductor tree branches. As always the numbering
starts with one and continues in sequence.

Directly from the circuit topology the following matrices can be formed.

AA is the vertex matrix defined previously.

E is the voltage source matrix defined previously.

G is the current source matrix defined previously.

\[ PE = [PE_{j,j}] \quad j = 1 \ldots, IB \]

PE\(_j\) is the resistance, capacitance or inductance of the jth block.

PE is referred to as the passive element matrix.

There are also the unknown matrices e and I, which are the output, as well as the state variable matrix

\[ x = \begin{bmatrix} \mathbf{e}_{C2} \\ \mathbf{J}_{L1} \end{bmatrix} \]

where \( \mathbf{e}_{C2} \) is the tree branch capacitor voltage matrix and \( \mathbf{J}_{L1} \) is the link inductor current matrix. It can be shown that for a circuit with IB blocks and NN nodes, the number of branches in any tree is \((IB - NN + 1)\).

The vertex matrix can be partitioned into two submatrices

\[ [\mathbf{AA}_{11} \quad \mathbf{AA}_{12}] \]

where \( \mathbf{AA}_{11} \) relates the nodes to the links and \( \mathbf{AA}_{12} \) relates the nodes to the tree branches. The fundamental cut
set matrix $Q$ can be formed by the relation $Q = [AA_{12}]^{-1} [AA]$ (12, p. 93-98). It can be shown that $Q = [-F' 1]$ where the prime (') refers to the transpose operation and $1$ is the unit matrix of dimension $NN \times NN$. $F$ can be partitioned as follows:

$$F = \begin{bmatrix}
F_{SC} & 0 & 0 \\
F_{RC} & F_{RG} & 0 \\
F_{LC} & F_{LG} & F_{LT}
\end{bmatrix}$$

where

- $F_{SC}$ topologically relates the capacitor links with the capacitor tree branches,
- $F_{RC}$ relates the resistor links with the capacitor tree branches,
- $F_{LC}$ relates the inductor links with the capacitor tree branches,
- $F_{RG}$ relates the resistor links with resistor tree branches,
- $F_{LG}$ relates the inductor links with resistor tree branches, and
- $F_{LT}$ relates the inductor links with inductor tree branches.

We can further note that the fundamental loop matrix is formed by $B = [1 \ F]$ where $1$ is an $IB \times IB$ unit matrix.

The passive element matrix must also be partitioned into capacitor, resistor, and inductor links (denoted by $C_1$, $R_1$, and $L_1$ respectively) and capacitor, resistor, and inductor tree branches (denoted by $C_2$, $R_2$, and $L_2$, respectively). That is,
The algorithm for forming the state equations was proposed by Bashkow and formalized by Bryant (2). This presentation of the algorithm is taken from Kuh and Rohrer (9) where it is developed in more detail.

Based on the particular ordering of the blocks, explained previously, and the submatrices, defined previously, the following matrices may be formed:

\[
\begin{bmatrix}
C_1 & 0 & 0 & 0 & 0 & 0 \\
0 & R_1 & 0 & 0 & 0 & 0 \\
0 & 0 & L_1 & 0 & 0 & 0 \\
0 & 0 & 0 & C_2 & 0 & 0 \\
0 & 0 & 0 & 0 & R_2 & 0 \\
0 & 0 & 0 & 0 & 0 & L_2 \\
\end{bmatrix}
\]

\[PE = \begin{bmatrix}
C_1 & 0 & 0 & 0 & 0 & 0 \\
0 & R_1 & 0 & 0 & 0 & 0 \\
0 & 0 & L_1 & 0 & 0 & 0 \\
0 & 0 & 0 & C_2 & 0 & 0 \\
0 & 0 & 0 & 0 & R_2 & 0 \\
0 & 0 & 0 & 0 & 0 & L_2 \\
\end{bmatrix}\]

\[C = C_2 + (F'_SC)(C_1)(F_SC)\]

\[L = L_1 + (F_LT)(L_2)(F_LT)\]

\[R = R_1 + (F_RG)(R_2)(F'_RG)\]

\[G = R_2^{-1} + (F'_RG)(R_1^{-1})(F_RG)\]

\[Y = (F'_RC)(R^{-1})(F_RC)\]

\[Z = (F_LG)(G^{-1})(F'_LG)\]

\[H = F'_LC - (F'_RC)(R^{-1})(F_RG)(R_2)(F'_LG)\]
Then the first of the matrices required for the state equations may be formed,

\[
A = \begin{bmatrix} C^{-1} & 0 \\ 0 & L^{-1} \end{bmatrix} \begin{bmatrix} -Y & H \\ -H' & -Z \end{bmatrix}
\]

(7)

The second matrix is formed in a similar manner.

\[
B = \begin{bmatrix} C^{-1} & 0 \\ 0 & L^{-1} \end{bmatrix}
\]

\[
\begin{bmatrix} F'_R C R^{-1} F R G R_2 & 0 & F'_S C_1 & F'_R C R^{-1} \\ 0 & -F_L G R^{-1} & -F L T L_2 & 0 & -F_L G R^{-1} F'_R G R_1 L_1^{-1} \end{bmatrix}
\]

(8)

The input vector is formed by

\[
u(t) = \begin{bmatrix} u_C \\ u_L \end{bmatrix}
\]

where

\[
u_C = (Q)(C)
\]

\[
u_L = (B)(E) \quad \text{(see reference (2))}
\]

The input vector may be partitioned by

\[
u_C = \begin{bmatrix} u_{C2} \\ u_{R2} \\ u_{L2} \end{bmatrix}
\]
and

\[ u_L = \begin{bmatrix} u_{C1} \\ u_{R1} \\ u_{L1} \end{bmatrix} \]

This algorithm provides the state equations for a linear passive circuit with no truncation error and a minimum of round-off error.

Once the state variables \( e_{C2} \) and \( J_{L1} \) and their derivatives have been found, the remaining element voltages and currents can be found from the following equations, (which may be developed from Kuh and Rohrer's Equation 39) (9).

**Tree branch capacitor current:**

\[ J_{C2} = C \frac{d}{dt} e_{C2} \]  \hspace{1cm} (9a)

**Link inductor voltage:**

\[ e_{L1} = L \frac{d}{dt} J_{L1} \]  \hspace{1cm} (9b)

**Tree branch inductor current:**

\[ J_{L2} = F' L T J_{L1} + u_{L2} \]  \hspace{1cm} (9c)
Link capacitor voltage:

\[ e_{C1} = -F_{SC} e_{C2} + u_{C1} \]  \hspace{1cm} (9d)

Tree branch and link resistor voltages:

\[
\begin{bmatrix}
e_{R1} \\
e_{R2}
\end{bmatrix}
= 
\begin{bmatrix}
1 & F_{RG}^{-1} \\
-F_{RG}(R_1)^{-1} & (R_2)^{-1}
\end{bmatrix}
\begin{bmatrix}
-F_{RC} e_{C2} + u_{R2} \\
+F_{LG} J_{L2} + u_{R1}
\end{bmatrix}
\]  \hspace{1cm} (9e)

Link and tree branch resistor currents:

\[ J_{R1} = e_{R1}(R_1)^{-1} \]  \hspace{1cm} (9f)

\[ J_{R2} = e_{R2}(R_2)^{-1} \]

Tree branch inductor voltage:

\[ F_{L1} e_{L2} = -F_{LC} e_{C2} - F_{LG} e_{R2} - e_{L1} + u_{L1} \]  \hspace{1cm} (9g)

Link capacitor current:

\[ F_{SC} J_{C1} = -F_{RC} J_{R1} - F_{LC} J_{L1} + J_{C2} - u_{C2} \]  \hspace{1cm} (9h)

Now the element voltage and element current vectors may be formed:
Then the block voltage matrix is given by \( e' = v_b - E \), and the block current matrix is given by \( I = J_b - C \). Since \( e' = AA'(e) \), \( e \) can be found by taking the inverse of a square submatrix of \( AA' \) and multiplying it by the corresponding submatrix of \( e' \). That is

\[
e = (AA_{12})^{-1}(e'_{\text{branches}}).
\]

To speed up the loading of the program or to increase the maximum size of the circuit which can be analyzed, the transient analysis program can be shortened by restricting the circuits to those containing neither inductor cut-sets nor capacitor loops. The various equations can be simplified because \( C_1 \) and \( L_2 \) are both equal to \( \phi \) (the null matrix) and \( F \) reduces to

\[
\begin{bmatrix}
F_{RC} & F_{RG} \\
F_{LC} & F_{LG}
\end{bmatrix}
\]
This shortened program should not appreciably change the running time.

**Matrix Inversion**

The formation of state variable circuit equations requires several matrices to be inverted. The algorithm used to make these inversions is given in Finkbeiner (4, p. 100-103).

**Solution to State Equations**

The state equations \( \frac{dx}{dt} = f([x], t) \) are equivalent to a system of linear first order differential equations. Consider the \( i \)-th equation.

\[
\frac{dx_i}{dt} = f_i([x], t) \tag{10}
\]

A simple numerical method for solving such an equation is found in Quinn's method, also known as Euler's modified method (8, p. 344-45). This method assumes that the values of \( x \) are known at some time, say \( t_N \). These values will be called \( x(N) \). The object is to find the value of \( x_i \) at time equal to \( t_N + \Delta t \), denoted by \( t_{N+1} \). From Equation 10, the value of \( \frac{dx_i}{dt} \) at time equal to \( t_N \) can be found. This value, \( f_i([x_j(N)], t_N) \), will be denoted by \( f_i(N) \). Then a predicted value for \( x_i \) at time equal to \( t_{N+1} \) may be found from the equation
This can be done for all the state equations. Now \( \frac{dx_i}{dt} \) may be found for the time \( t_{N+1} \) (using Equation 10 and \( x = [P(x_i)] \)). Finally a corrected value for \( x_i(N+1) \) can be found using the equation

\[
P(x_i)_{N+1} = x_i(N) + (\Delta t)(f_i(N))
\]

Again this can be done for all the equations in the system. While a second corrected value can in turn be computed, in order to minimize round-off error, the computations are usually stopped after the first corrected values have been found. Thus \( x(N+1) = [C(x_i(N+1))] \). The computations start at time equal to zero and, in this program, the state variables are assumed to be zero initially. Thus the inductors and capacitors may not have an initial current or voltage. However, the program can easily be modified to accept initial conditions should such ever be required.

### Round-off and Truncation Error in Solving the State Equations

When numerical methods are used, it is necessary to consider possible error. The truncation error per step for Quinn's method is proportional to \( (\Delta t)^3 \frac{d^3x}{dt^3} \) and will decrease as \( \Delta t \) is made smaller. Truncation is not the only source of error however. Round-off
error can affect the solution and will tend to increase with the number of iterations. An examination of the equations used in Quinn's method shows that the method is relatively stable (6, p. 186-192). That is, the error relative to the magnitude of the solution either decreases or remains constant.

The value of $\Delta t$ used in a transient analysis must be a compromise between two conflicting criteria. A large value of $\Delta t$ would cause a high truncation error, while a smaller value would increase the running time and the round-off error. The ideal value for $\Delta t$ will depend upon the state equations being solved. In general $\Delta t$ should be small enough to keep truncation error within reasonable bounds but in any case should be no smaller than necessary.

As a test the equation $\frac{dx}{dt} = -x + u(t)$ was solved by the computer, using Quinn's method. This equation has an exact solution $x = u(t) - e^{-t}$. The value of $x$ at time equals four seconds can be found to be 0.981684, rounded to six places. By using a time step of $\Delta t$ equals 0.1 seconds, $x$ was computed to be 0.98155229 at time equals four seconds. With a smaller value for $\Delta t$, 0.005 seconds, $x$ was computed to be 0.98168409. The method appears to be quite accurate considering its simplicity.

**Sources**

Special attention must be paid to the way sources are specified.
In the program, each source consists of a generator and a multiplier. The value of the source as used in the analysis is the product of the magnitude of the generator (given by the generator definition cards) and the magnitude of the multiplier (given by the block definition cards).

Provisions have been made for five types of generators which have the following characteristics.

FUNCTION GENERATORS--PERIODIC OR NON-PERIODIC:
The function generator consists of up to twenty values of time and corresponding function values. For any given time input, a value for the output (the function) is determined by linear interpolation. The Non-periodic function generator holds the last value of the function for all times outside the defined function's time domain. On the other hand the Periodic function generator will start over at time equal to zero when the input exceeds the defined time domain. Thus the function has a period equal to the defined domain.

SINE WAVE GENERATOR: This generator will produce a sine wave of variable magnitude and period.

SQUARE WAVE GENERATOR: This generator will produce a square wave which starts at a value of plus or minus its magnitude and which changes sign everytime the input time reaches one half the period.

STEP FUNCTION GENERATOR: This generator has a value
equal to zero if the input time is less than the delay specified. When time reaches the delay time, the generator assumes a constant value equal to its magnitude and maintains that value for all subsequent times.

**Input Format**

The same basic input format is used for the transient analysis as for the a-c and d-c analyses. One major change is that the value of the voltage and current sources is the product of the value given on the block specification card and the magnitude given in the generator specification cards. A special way of numbering the block, discussed on page 27, must be used. And finally no dependent sources are allowed.

The first card specifies the size of the circuit and contains three integer fields:

```
IB  NN  0
```

IB is the number of blocks and may not exceed nine.

NN is the number of nodes (excluding the reference nodes which are labeled zero). NN must be less than nine.

0 is the character zero. This is the number of dependent sources allowed which for this version is zero.
The next 1B cards describe the blocks of the circuit. It is assumed that the blocks have been numbered in the correct sequence. Each card contains three integers followed by four floating point fields in the following format:

\[ \text{I J K R RR D B} \]

In the above,

- I is the block number.
- J is the source node number.
- K is the sink node number.
- R is the block resistance.
- RR is the block reactive element. If the block contains a capacitor, RR is negative and gives the capacitance in farads. If the block contains an inductor, RR is positive and gives the inductance in henries. Either RR or R (but not both) must be zero.
- D is the voltage source multiplier magnitude (in volts).
- B is the current source multiplier magnitude (in amps).

The generator specification cards follow the block specification cards. The generators are entered in the order of their block number. If a block contains both a voltage source and a current source, the corresponding voltage generator is specified first. A circuit may
have a maximum of five sources, and only one of these may be a function generator. The generators are specified as follows:

1. Non-periodic Function Generator:
The first card contains the four characters FUNC.
The second card contains the number of points in the function. Up to 20 points may be specified.
The next cards each contain two floating point fields and specify the function. The first field gives the values of time and the second field gives the value of the function at that time. Unless otherwise specified the function is assumed to be zero at time equals zero.

2. The Periodic Function Generator is specified by the same card sequence as the non-periodic generator except that the first card must contain the four characters FUNP.

3. Sine Wave Generator is specified by two cards:
The first card contains the four characters SINE.
The second card contains two floating point fields. The first field gives the zero to peak magnitude of the wave, and the second gives the period of the wave (in seconds).

4. The Square Wave Generator is specified by two cards:
The first card contains the four characters SQUA.
The second card contains two floating point fields which give the zero to peak magnitude of the wave and the
period of the wave in seconds.

5. The Step Function Generator is specified by two cards:
The first card contains the four characters \texttt{STEP}.
The second card contains two floating point fields, the first
of which gives the magnitude of the step and the next
gives the delay time in seconds.

6. The zero generator (one which has an output of zero for all
time) is specified by a first card which contains some
character set other than \texttt{FUNC, FUNP, SINE, SQUA,}
or \texttt{STEP}. Whenever a zero generator is called, the
computer goes to manual mode. This specification
could occur if one of the previously listed generators had
been incorrectly called on its first card. This may be
corrected using sense switch one and re-entering the
corrected cards. On the other hand the zero specifcation
may be correct in which case the program can be
continued.

The final card, containing three floating point fields, specifies
the times needed in the numerical solutions:

\[
\begin{array}{c}
\text{DT} \\
\text{PI} \\
\text{FT}
\end{array}
\]

\text{DT} \text{ is the value of } \Delta t \text{ in seconds to be used by the numerical in-}
tegration.
PI gives the output interval.

FT gives the final time. The voltages and currents will be punched (or printed) at time equals zero seconds, PI seconds, 2PI seconds, and so on until time equals or exceeds FT.

**Operation**

This program is divided into three passes, with passes two and three called automatically. Because of this the entire program must be loaded for each circuit to be analyzed. The data cards are inserted between pass one and pass two. Passes one and two form the state equations, and pass three solves these equations. The output is punched on cards and at the user's option may be typed. (Sense switch three on causes the output to be typed.) When the computer finishes, the last computed values of the voltages and currents are always typed out. After an examination of the results, it is possible to extend the computational period.

Operational procedure is similar to that used for batch processing of a-c circuits.
Figure 4. Transient analysis example.

Figure 4a. Bridge-tee circuit (7, p. 94-98).

Figure 4b. Input to transient analysis program.

```
7 3 0
1 1 0 1000. 0. 0. 02
2 3 0 1. E3 0. 0. 0.
3 1 2 0. 1. 0. 0.
4 2 3 0. 1. 0. 0.
5 1 3 0. -1. E-6 0. 0.
6 2 0 0. -1. E-6 0. 0.
7 3 0 0. -1. E-6 0. 0.
STEP
1. 0.
1. E-5 50. E-5 2. E-3
```

The state variables *x* are the voltages across *C*₅, *C*₆, and *C*₇ and the currents through *L*₃ and *L*₄ in that order.
TRANSIENT ANALYSIS

45

Figure 4c. Computer output for circuit shown in Figure 4a.
<table>
<thead>
<tr>
<th></th>
<th>(1.00427867E-02)</th>
<th>(1.8447287E-01)</th>
<th>(1.04457796E-01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C 1</td>
<td>(-.94601938E-02)</td>
<td>(2.1923970E-02)</td>
<td>(2.5070047E-03)</td>
</tr>
<tr>
<td>C 2</td>
<td>(1.0445736E-02)</td>
<td>(6.579360E-03)</td>
<td>(8.9416935E-02)</td>
</tr>
<tr>
<td>C 3</td>
<td>(-.93082433E-02)</td>
<td>(2.3745495E-02)</td>
<td></td>
</tr>
</tbody>
</table>

\(T = 2.0000E-02, \text{ SEC}\)

<table>
<thead>
<tr>
<th></th>
<th>(1.00427859E-02)</th>
<th>(2.3738611E-02)</th>
<th>(2.3738622E-02)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C 1</td>
<td>(-.9306075E-02)</td>
<td>(3.176170E-03)</td>
<td>(2.3745495E-02)</td>
</tr>
<tr>
<td>C 2</td>
<td>(2.3738622E-02)</td>
<td>(3.3082433E-02)</td>
<td>(2.8745495E-02)</td>
</tr>
</tbody>
</table>

Figure 4c. Continued.
CONCLUSIONS

The programs have been debugged but have not been used extensively. Changes suggested by continued use will probably be limited to input/output of data or to error indication and correction.

The limited size of the memory has dictated that many features of ECAP be omitted. The two most serious deficiencies are the inability to handle non-linear circuit elements and the lack of controlled sources in the transient analysis. In ECAP non-linear circuits are simulated using switches, while active circuits are simulated using controlled sources. These two elements may not, however, provide the best possible simulation; the state equations may suggest better ways to approach the problem. Programs to analyze non-linear and/or active circuits should probably be more closely tailored to the specific circuit than is possible with ECAP.

The machine also limits the accuracy of the solution. With PDQ it is possible to get no more than eight digits. For almost all circuits the solution will not be that accurate. The last three digits can usually be rounded-off. The error will depend upon the size of the circuit and range of magnitudes of the impedance. For transient analysis, the integration time step and the number of iterations which occur in the numerical solution of the state equations also affect the accuracy.
The answers computed by these programs compared favorably with those computed by ECAP. In most cases they agree when rounded to three places; however, it should be remembered that ECAP can only approximate the reactive elements in the transient equations while the state equations are exact. Thus ECAP cannot be taken as an exact standard for transient analysis. Until more experience is gained with these programs it is necessary that a user look critically at all solutions and not take them as being correct to an arbitrary number of digits.

Another disadvantage in using the 1620 is that it is relatively slow. For small circuits of about five blocks, and for small integration time steps of about 0.1 seconds the analysis is fast. But as the size of the circuit increases and as $\Delta t$ is made smaller, the time required to make the analysis increases rapidly. The amount of time required to load the programs will be a factor only when a single small circuit is being analyzed. For batch processing or for transient analysis with a small $\Delta t$, less than 0.01 seconds, the loading time is not too significant when compared with the time required for the analysis.

The d-c analysis used as an example took about one minute on the 1620, the a-c analysis took about five minutes, while the transient analysis took about 30 minutes. A faster computer such as the CDC 3300 would not require as much time but would be more expensive.
per unit of time. It would probably be cheaper to run the larger problems on the faster machine. However, the 1620 allows the user directly to control the program and is more accessible than the 3300.

It would appear that the 1620 programs will probably be used for instructional purposes. As the programs are used, they may be modified or expanded as dictated by needs and experience.
BIBLIOGRAPHY


APPENDICES
APPENDIX I

List of Symbols Used in the D-C Analysis Program

A(I, J) is the vertex matrix.
Y(I, J) is the block admittance matrix.
E(I) is the block voltage source matrix.
C(I) is the block current source matrix.
T(I), TT(I, J), S(I) are used as temporary storage for matrix operations.

LL is the circuit number.
IB is the number of blocks.
NN is the number of nodes.
IC is the number of dependent current sources. IC also indicates whether dep source data is entered by card or by typewriter.

I, J, K are used in input statements and as indices in matrix operations.

KE is the block number, used as an index to input block specification cards.

L is an error flag

N indicates whether block data is being entered by card or by typewriter.

PP, D, B are used as temporary storage in input statements. PP is also used for temporary storage in computations.
D-C Analysis

Start

51

Initialize

68

Read Block Data Card

Yes

Error

No

Form A_{i,j}; Y_{i,j}; E_{i,j} C_{i,j} from data

76

Card Input

Flag

Typewriter Input

Read Next Card

I

94

IC

92

88

All Cards Read

Read Dependent Source Card

96

94

Type "Type Input"

110

112

ON

OFF

SS1

52

470
D-C Analysis

200

T = C_i Y_{i,j}^T *E_j

204

TT_{j,k} = A_{j,i}^T Y_{i,j} * A_{j,k}

212

S = T_{j,i} * A_{j,i}

218

S = \text{Solve the System}\n\quad (TT_{j,k} * e_k = S_j)

228

Output $S_k$,\nnode-to-reference voltage

231

T_i = E_i - A_{i,j} * S_j

235

S_i = C_i + Y_{i,j} * T_{i,j}

244

3

Error

No

Yes

Error in Data

No Run

306

255
D-C Analysis

Output $S_k$, the Block Currents

On

SS3

Type "SS1 on to Type Input"

Pause

Off

252

Type "SS2 on to Correct Error"

100

Type "Next Circuit"

100

Accept Data from Console Typewriter

102

On

SS2

104

Off

102

Block Data

76

Dependent Source Data

106

Dependent Source Controlled by its own Block

92

Zero Element in Y Matrix

J-K

≠ 0

= 0

92

51
**LINEAR, D-C CIRCUIT ANALYSIS**
**STEADY STATE NODE VOLTAGES AND BRANCH CURRENTS**
**CARD INPUT/OUTPUT**
**OPTIONAL CONSOLE TYPEWRITER INPUT/OUTPUT**

**INPUT SPECIFICATION - PDQ FREE FORMAT**

**FIRST CARD - NO. OF BLOCKS - MAX. OF 20**

- NO. OF NODES (EXCLUDING REFERENCE NODES WHICH ARE GIVEN THE NUMBER ZERO) - MAX. OF 19
- NO. OF DEPENDENT CURRENT SOURCES

**ONE CARD FOR EACH BLOCK, STARTING WITH BLOCK 1**
**CONTINUING IN SEQUENCE**
- BLOCK NO.
- SOURCE NODE NO.
- SINK NODE NO.
- BLOCK RESISTANCE (IN OHMS)
- VALUE OF VOLTAGE SOURCE (IN VOLTS)
- VALUE OF INDEPENDENT CURRENT SOURCE (IN AMPS)

**ONE CARD FOR EACH DEPENDENT CURRENT SOURCE**
- THE CHARACTER ZERO
- NO. OF CONTROLLING BRANCH
- NO. OF BRANCH HAVING SOURCE
- VALUE OF BETA FOR SOURCE

**EXAMPLE OF FORM ---**

```
  3 2 1
  1 0 1 100  10  0
  2 1 2 1000  0  1
  3 2 0 10E4  0  0
  0 1 2 1
```

**SENSE SWITCH SETTINGS**

**SS1** - IF ON AT END OF COMPUTATION CYCLE, WILL ALLOW MODIFICATION OF CIRCUIT CURRENTLY IN MEMORY. TO RETURN TO COMPUTATION MODE TURN SS1 OFF PRIOR TO LAST TYPED INPUT. IF OFF AT END OF COMPUTATION CYCLE, COMPUTER WILL EXPECT CARD INPUT.

**SS2** - IF ON AFTER TYPED INPUT, THE INPUT IS IGNORED AND A CORRECTED, TYPED INPUT IS ACCEPTED. IF OFF, TYPED INPUT PROCESSED.
SS3 - IF ON, CARD INPUTS AND OUTPUTS ARE TYPED ON CONSOLE TYPEWRITER. IF OFF, OUTPUT IS ON CARDS ONLY.

* FOR BATCH PROCESSING - ALL SENSE SWITCHES OFF.

* FOR ONLINE PROCESSING - SS3 ON.
  SSI ON TO ACCEPT CIRCUIT MODIFICATIONS.
  SSI OFF TO COMPUTE.
DIMENSION A(20,19), Y(20,20), E(20), C(20)
DIMENSION T(20), IT(19,19), S(20)
PRINT 174
PUNCH 174
LL = 0
51 READ 150, IB, NN, IC
   LL = LL + 1
   PUNCH 169, LL
   IF (IB - 20) 52, 52, 300
52 IF (NN - 19) 54, 54, 300
54 I = I + 1
56 J = I
58 A(I,J) = 0.
   Y(I,J) = 0.
   J = J + 1
   IF (J - 18) 58, 58, 64
64 I = I + 1
   IF (I - 13) 56, 56, 66
66 L = 0
67 KE = 0
   N = 0
68 READ 151, I, J, K, PP, D, B
   IF (SENSE SWITCH 3) 69, 71
69 PRINT 156, I, J, K, PP, D, B
   IF (PP) 70, 101, 70
70 IF (I) 72, 303, 72
72 KE = KE + 1
   IF (KE - 1) 303, 74, 303
74 IF (L) 56, 56, 86
76 PUNCH 178, I, J, K, PP, D, B
77 IF (J) 78, 80, 78
78 A(I,J) = 1.
80 IF (K) 82, 84, 82
82 A(I,K) = -1.
84 Y(I,J) = 1./PP
   T(I) = 0
   C(I) = 0.
   IF (I) 110, 86, 110
85 IF (KE - 1) 88, 94, 94
88 IF = IF - 1
   READ 152, I, J, K, PP
   IF (SENSE SWITCH 3) 89, 91
89 PRINT 157, I, J, K, PP
91 IF (I) 94, 94, 94
92 IF (I) 94, 94, 94
92 PUNCH 179, J, K, PP
95 Y(K,J) = Y(I,J) + PP*Y(J,J)
94 IF (I) 110, 96, 96
96 IF = IF + 1.
100 PRINT 160
   N = 1
IC = -1
102 ACCEPT 151, I, J, K, PP, D, B
   IF (SENSE SWITCH 2) 102, 104
104 IF (I) 76, 106, 76
106 IF (J - K) 108, 92, 108
108 Y(I, J) = 0.
   GO TO 92
110 IF (SENSE SWITCH 1) 112, 200
112 PRINT 199
   GO TO 102
200 I = 1
201 T(I) = C(I)
   J = 1
203 T(I) = T(I) - Y(I, J) * S(J)
   IF (I) 202, 202, 203
203 I = I + 1
204 K = 1
205 I = 1
206 PP = 0.
   J = 1
207 PP = PP + Y(I, J) * A(J, K)
   J = J + 1
209 J = 1
210 T(J, K) = T(J, K) + PP * A(I, J)
   J = J + 1
216 I = I + 1
219 ? = K + 1
220 I = 1
221 S(I) = 0.
   J = 1
216 S(I) = S(I) + T(J) * A(J, I)
   J = J + 1
217 I = I + 1
219 I = ?
223 T(I) = T(I) / PP
   J = I
222 T(I, J) = T(I, J) / PP
   J = J + 1
223 K = 1
224 IF (K - 1) 229, 226, 229
225 PP = TT(K,J)
     S(K) = S(K) - S(I)*PP
     J = J + 1
226 TT(K,J) = TT(K,J) - T(I,J)*PP
     J = J + 1
227 I = I + 1
     IF (! - MN) 220, 220, 230, 230
228 PUNCH 170
     I = I + 1
229 IF (SENSE SWITCH 3) 270, 271
230 PRINT 165, I, S(I)
231 PUNCH 169, I, S(I)
     I = I + 1
232 T(I) = E(I)
     J = 1
233 T(I) = T(I) + A(I,J)*S(J)
     J = J + 1
234 I = I + 1
     IF (! - IB) 241, 241, 244
235 I = I + 1
236 C(I) = C(I) + S(I) + Y(I,J)*T(J)
     J = J + 1
237 I = I + 1
     IF (! - IB) 241, 241, 244
238 244 PUNCH 172
     I = I + 1
239 IF (SENSE SWITCH 3) 272, 273
240 PRINT 171, I, S(I)
241 PUNCH 171, I, S(I)
     I = I + 1
     IF (! - IB) 251, 251, 252
242 IF (SENSE SWITCH 3) 253, 254
243 PRINT 153
244 PUNCH 153
245 IF (SENSE SWITCH 1) 100, 255
246 PRINT 153
     GO TO 31
247 PRINT 180
     I = I
     GO TO 61
248 PRINT 161, I
L = 1
GO TO 70
203 PRINT 155
L = 1
GO TO 96
204 PRINT 155
L = 1
GO TO 94
205 PRINT 173, I
GO TO 206
206 PRINT 154
PUNCH 154
GO TO 255

150 FORMAT (3I3)
151 FORMAT (3I3, 3E15.8)
152 FORMAT (3I3, E15.8)
153 FORMAT (9HNEXT CIRCUIT)
154 FORMAT (2OH INPUT ERROR, NO RUN)
155 FORMAT (18HCARDS OUT OF ORDER)
156 FORMAT (3I3, 3E12.9)
157 FORMAT (3I3, E12.9)
158 FORMAT (20HSS1 ON TYPE INPUT)
159 FORMAT (10HTYPE INPUT)
160 FORMAT (34HTYPE INPUT, SS2 ON TO CORRECT ERROR)
161 FORMAT (2HR(I, I2, 3H)=0)
162 FORMAT (11HERROR CHECK)
163 FORMAT (12HREP OR PAUSE)
164 FORMAT (13H, I3, ZX, ...15.8)
165 FORMAT (/7I3X, 7HCHCT1, 13)
166 FORMAT (/72H I VOLTAGE I1-I0, VOLTS)
167 FORMAT (11H1, I3, ZX, E15.8)
168 FORMAT (11H1, I3, ZX, E15.8)
169 FORMAT (/74H I 1ST BRANCH CURRENT, AMPS)
170 FORMAT (6CHAMP(A, I2, 5H)=0)
171 FORMAT (/77H1 D-C CIRCUIT ANALYSIS)
172 FORMAT (2H R1, (13,9H), K(I,13,7H) TO R4,13,9H), K=I1.8,
173, E=C15.8, 3H(I,13,9H)=0)
174 FORMAT (2H DCP, B1,13,13H) CONTROLS B(I,13,9H), BETA =
175 (13H)
176 FORMAT (14HCIRCUIT OVERSIZED)
END
APPENDIX II

List of Symbols Used in A-C Analysis Program

A(I, J) is the vertex matrix.

YR(I, J) is the block admittance matrix (real component).

YI(I, J) is the block admittance matrix (imaginary component).

E(I) is the block voltage source matrix.

C(I) is the block current source matrix.

RTS(I) is the block reactive element matrix.

TR(I), TI(I), TTR(I, J), TTI(I, J), SR(I), SI(I) are used as temporary storage for matrix operations.

LL is the circuit number.

IB is the number of blocks.

NN is the number of nodes.

IC is the number of dependent current sources. IC also indicates whether dep source data enter by card or typewriter.

I, J, K are used in input statements and as indices in matrix operations.

KE is the block number, an index used to input block specification cards.

N indicates either card input or typewriter input.

L is an error flag.

WW is the initial frequency.
DW is the frequency incrementation.

WX is termination frequency.

R, RR, D, B are used as temporary storage in input statements.

Also used as temporary storage for matrix computations.
A-C Analysis

Start

400
Initialize

415
Read Frequency Card
WW, DW, WX

418

DW ≠ 0

DW = 0

DW = 1000.

Card Input

430
Type Input

504

Card Input

434

Read Block Data Card

442

Error

Yes

No

444

Form \( YR_{i,j}; \)
\( Y_{i,j}; A_{i,j}; E_{i}; C_{i} \)

Typed Card

Flag

Card Input

448

511

464

Read Next Card

481
Block Data Cards Read
A-C Analysis

481

IC

Last Card Read

470

Read Dependent
Current Card

474

Form Entry in
YR and YI

Card Input

Flag

Typed
Input

511

Error No Run

400

Yes

Error

482

No

400

600

600

TR \rightleftharpoons C \_YR \_YR \_E

TI \_YI \_YI \_E

608

TTR \_A \_YR \_A

TTI \_A \_YI \_A

624

SR \_TR \_A

Si \_TI \_A

632
A-C Analysis

1. Solve the System
   \((TTR_j, k) e_k = SR_j\)
   \((TTI_j, k) e_k = SI_j\)

2. Solution \(e_k\) Placed in \(SR_j\) (real part) and \(SI_j\) (imaginary)

3. Output Frequency

4. Convert to Polar Co-ordinates
   \(\theta \leftarrow \arctan \left(\frac{SI_j}{SR_j}\right)\)
   \(M_j \leftarrow SR_j / \cos \theta\)

5. Key
   - Output Voltage \(M_j \angle \theta_j\)
   - Output Currents \(M_j \angle \theta_j\)
A-C Analysis

690

\[ TR_i \leftarrow A_i \cdot SI_{i,j} \]
\[ TI_i \leftarrow E_i + A_i \cdot SR_j \]

690

\[ SR_i \leftarrow C_i \cdot YR_{i,j} \cdot TR_j \]
\[ SI_i \leftarrow YR_{i,j} \cdot TR_j \]

702

\[ SR_i \leftarrow SR_i + YR_{i,j} \cdot TI_j \]
\[ SI_i \leftarrow SI_i + YR_{i,j} \cdot TI_j \]

710

656

720

\[ W:WX \]

740

On

SS3

Off

742

Type "SS1 on to type input"

744

Pause

600

Increment Frequency

724

722

<

744

>
A-C Analysis

744

746 Off SS1 On 500

Type "Next Circuit"

400

Print "Type Input (Frequency). SS2 on to Correct Error"

502

Accept Data

On SS2 Off 418

Set YI, j to Initial Value

504

512

Print "Type Data (Param)"

Accept Data

On SS2 Off 514

518

J-K

516 Independent Source

I Block Data 444

Self Dependent Source

478

YR, k, j \approx 0.

YI, k, j \approx 0.
LINEAR A-C CIRCUIT ANALYSIS
STEADY STATE NODE VOLTAGES AND BRANCH CURRENTS
CARD INPUT/OUTPUT
OPTIONAL CONSOLE TYPEWRITER INPUT/OUTPUT

* INPUT SPECIFICATION — RDD FREE FORMAT

FIRST CARD - NO. OF BLOCKS - MAX. OF 14
NO. OF NODES (EXCLUDING REFERENCE NODES
WHICH ARE GIVEN THE NUMBER ZERO)
- MAX. OF 13
NO. OF DEPENDENT CURRENT SOURCES.

SECOND CARD SPECIFIES FREQUENCY RANGE —
INITIAL FREQUENCY (IN HERTZ)
INCREMENTAL FREQUENCY (MULTIPLICATIVE)
FINAL FREQUENCY.

ONE CARD FOR EACH BLOCK, STARTING WITH BLOCK 1
CONTINUING IN SEQUENCE —
BLOCK NO.
SOURCE NODE NO.
SINK NODE NO.
BLOCK RESISTANCE (IN OHMS)
BLOCK CAPACITANCE (NEGATIVE NUMBER) IN
FARADS, OR BLOCK INDUCTANCE (POSITIVE
NO.) IN HENRIES
VALUE OF VOLTAGE SOURCE (IN VOLTS)
VALUE OF INDEPENDENT CURRENT SOURCE
(IN AMPS).

ONE CARD FOR EACH DEPENDENT CURRENT SOURCE —
THE CHARACTER ZERO
NO. OF CONTROLLING BRANCH
NO. OF BRANCH HAVING SOURCE
VALUE OF BETA FOR SOURCE.

EXAMPLE OF FORM ——

```
   3  2  1
   1  0  1  100.  -5.6-6  10.  0.
   2  1  2  0.  6.6-3  0.  1.0
   3  2  0  10.64  0.  0.  0.
   0  1  2
```
*SENSE SWITCH SETTINGS -*

- **SS1** - If on at end of computation cycle, will allow modification of parameters of circuit currently in memory. To return to computation mode turn SS1 off prior to last typed input. If off at end of computation cycle, computer will expect card input.

- **SS2** - If on after typed input, the input is ignored and a corrected, typed input is accepted. If off, typed input processed.

- **SS3** - If on, card inputs and outputs are typed on console typewriter. If off, output is on cards only.

*FOR BATCH PROCESSING - ALL SENSE SWITCHES OFF.*

*FOR ON-LINE PROCESSING - SS3 ON.*

SS1 on to accept circuit modifications.

SS1 off to compute.

*SPECIAL RESTRICTIONS -*

- If circuit is to be analysed using frequency incrementation, a block should contain no more than one dependent source.

- If circuit is to be analysed using parameter modification, controlling blocks should contain neither capacitors or inductors.

- If the resistance in a controlling block is changed, every source controlled by that block must be subsequently reentered.
**DIMENSION** N(14), C(14)
**DIMENSION** A(14,13), YR(14,14), YI(14,14)
**DIMENSION** T(14), T1(14), T(13,13), T1(13,13)
**DIMENSION** S(14), S(14)

PRINT 171
PUNCH 171

LL = 0
400 READ 150, IB, NN, IC
402 LL = LL + 1
   PUNCH 169, LL
IF (IB = 14) 404, 404, 300
404 IF (NN = 13) 406, 406, 300
406 I = I + 1
408 J = 1
410 A(I,J) = 0.
   YR(I,J) = 0.
   YI(I,J) = 0.
   J = J + 1
   IF (J = IB) 410, 410, 412
412 I = I + 1
   IF (I = IB) 408, 408, 414
414 L = 0
415 KE = 0
416 READ 162, WW, DW, WX
418 WW = WW * 6.2831853
   W = WW
   WX = WX * 6.2831853
   IF (WW) 420, 420, 430
420 WW = 1000.
   PRINT 166
420 IF (I) 504, 504, 504
434 READ 181, !, J, K, R, RR, D, B
   IF (SEND SWITCH 3) 436, 436
436 PRINT 184, I, J, K, R, RR, D, B
438 IF (I) 440, 440, 440
440 KE = KE + 1
   IF (KE = I) 303, 442, 303
442 IF (L) 464, 444, 464
444 IF (R) 444, 444, 444
446 IF (RR) 450, 301, 452
448 YR(I!,J!) = 1/K
   IF (RR) 450, 454, 454
450 YI(I!,J!) = -RR*W
   GO TO 454
452 YI(I!,J!) = -1./RR*W
454 PUNCH 178, I, J, K, R, RR, D, B
   IF (J) 456, 456, 456
456 A(I,J) = 1.
458 IF (Y) 460, 460, 460
460 A(I,J) = -1.
462 (IY) = 1
\( C(I) = \# \)
\( \text{RTS}(I) = RR \)
\( \text{IF} \ (N) \ 511, 464, 511 \)
\( 464 \ \text{IF} \ (KE = IB) \ 434, 481, 481 \)
\( 470 \ IC = IC - 1 \)
\( \text{READ} \ 152, I, J, K, R \)
\( \text{IF} \ (\text{SENSE SWITCH} \ 3) \ 472, 474 \)
\( 472 \ \text{PRINT} \ 151, I, J, K, R \)
\( 474 \ \text{IF} \ (I) \ 304, 476, 304 \)
\( 476 \ \text{IF} \ (L) \ 481, 478, 481 \)
\( 478 \ \text{PUNCH} \ 179, J, K, R \)
\( 481 \ \text{YR}(K,J) = \text{YR}(K,J) + R*\text{YR}(J,J) \)
\( \text{YI}(K,J) = \text{YI}(K,J) + R*\text{YI}(J,J) \)
\( 481 \ \text{IF} \ (IC) \ 511, 402, 470 \)
\( 482 \ \text{IF} \ (L) \ 306, 600, 306 \)
\( 500 \ \text{PRINT} \ 159 \)
\( N = 1 \)
\( 502 \ \text{ACCEPT} \ 162, WW, DW, WX \)
\( \text{IF} \ (\text{SENSE SWITCH} \ 2) \ 502, 418 \)
\( 504 \ J = 1 \)
\( 505 \ \text{IF} \ (\text{RTS}(J)) \ 506, 510, 508 \)
\( 506 \ \text{YI}(J,J) = -\text{RTS}(J)*W \)
\( \text{GO} \ \text{TO} \ 510 \)
\( 508 \ \text{YI}(J,J) = -1/(\text{RTS}(J)*W) \)
\( 510 \ J = J + 1 \)
\( \text{IF} \ (J - J) \ 505, 505, 511 \)
\( 511 \ \text{IF} \ (\text{SENSE SWITCH} \ 1) \ 512, 600 \)
\( 512 \ \text{PRINT} \ 160 \)
\( \ \text{ACCEPT} \ 181, I, J, K, R, RR, D, B \)
\( \text{IF} \ (\text{SENSE SWITCH} \ 2) \ 512, 514 \)
\( 514 \ \text{IF} \ (I) \ 444, 516, 444 \)
\( 516 \ \text{IF} \ (J - K) \ 518, 476, 510 \)
\( 518 \ \text{YR}(K,J) = 0. \)
\( \text{YI}(K,J) = 0. \)
\( \text{GO} \ \text{TO} \ 478 \)
\( 600 \ ! = 1 \)
\( 602 \ \text{TR}(I) = 0. \)
\( \text{TR}(I) = C(I) \)
\( J = 1 \)
\( 604 \ \text{TR}(I) = \text{TR}(I) - \text{YR}(I,J)*E(J) \)
\( \text{YI}(I) = \text{YI}(I) + \text{YI}(I,J)*E(J) \)
\( \text{TR}(I,J) = 0. \)
\( \text{YI}(I,J) = 0. \)
\( J = J + 1 \)
\( \text{IF} \ (J - I) \ 604, 604, 606 \)
\( 606 \ ! = 1 + 1 \)
\( \text{IF} \ (! - IB) \ 602, 602, 608 \)
\( 608 \ ! = 1 \)
\( 610 \ ! = 1 \)
\( 612 \ ! = 0. \)
\( C = 1 \)
\( J = 1 \)
614 \( R = R + \gamma R(I,J)*A(J,K) \)
\( RR = RR + Y(I,J)*A(J,K) \)
\( J = J + 1 \)
IF (J - IB) 614, 614, 616
616 J = 1
618 TTR(J,K) = TTR(J,K) + k*A(I,J)
\( TTI(J,K) = TTI(J,K) + RR*A(I,J) \)
\( J = J + 1 \)
IF (J - NN) 615, 615, 620
620 I = I + 1
IF (I - IB) 612, 612, 613
622 K = K + 1
IF (K - NN) 610, 610, 624
624 I = 1
626 SR(I) = 0,
\( SI(I) = 0 \)
628 SR(I) = SR(I) + TR(J)*A(J,I)
\( SI(I) = SI(I) + TI(J)*A(J,I) \)
\( J = J + 1 \)
IF (J - IB) 628, 628, 630
630 I = I + 1
IF (I - NN) 626, 626, 632
632 I = 1
634 IF (TTR(I,I)) 638, 638, 638
636 IF (TTI(I,I)) 638, 638, 638
638 D = TTR(I,I)*TTR(I,I) + TTI(I,I)*TTI(I,I)
\( P = TTR(I,I)/D \)
\( RR = -TTI(I,I)/D \)
\( D = SR(I) \)
\( SR(I) = D*R + SI(I)*RR \)
\( SI(I) = D*RR + SI(I)*R \)
640 I = TTR(I,J)
\( TTR(I,J) = D*R - TTI(I,J)*R \)
\( TTI(I,J) = D*RR + TTI(I,J)*R \)
\( J = J + 1 \)
IF (J - NN) 640, 640, 642
642 \( R = \) 
644 IF (K - 1) 646, 646, 646
646 P = TTR(K,I)
\( PP = TTR(K,I) \)
\( CR(K) = CR(K) - (SR(I)*R - SI(I)*RR) \)
\( CI(K) = CI(K) - (SR(I)*RR + SI(I)*R) \)
649 TTR(K,J) = TTR(K,J) - (P*K)*R - TTI(K,J)*R
\( TTI(K,J) = TTI(K,J) - (K*I)*R + TTI(K,J)*R \)
\( J = J + 1 \)
IF (J - NN) 648, 648, 650
650 \( V = V + 1 \)
IF (V - NN) 644, 644, 652
652 I = I + 1
IF (I - NN) 634, 634, 654
654 PUNCH 165, W
PUNCH 170
PUNCH 182
J = 0
IF (SENSE SWITCH 3) 655, 656
655 PRINT 164, W
656 I = 1
658 K = 0
660 R = SR(I)
RR = R(I)
IF (R(I) 668, 662, 670
662 IF (RR) 664, 660, 664
664 R = RR
D = 0.
GO TO 675
666 B = -RR
D = -R0.
GO TO 675
668 K = 1
R = - R
670 D = RR/R
D = ATAN(D)
B = R/COS(D)
D = 0.5729578E2
672 IF (J) 675, 675, 674
674 D = 180. - D
675 IF (J) 676, 676, 682
676 PUNCH 168, I, B, D
IF (SENSE SWITCH 3) 678, 680
678 PRINT 168, I, B, D
680 I = I + 1
IF (I < 10) 658, 658, 690
682 PUNCH 171, I, B, D
IF (SENSE SWITCH 3) 684, 686
684 PRINT 171, I, B, D
686 I = I + 1
IF (I < 10) 658, 658, 720
690 I = 1
692 TR(I) = 0
TR(I) = 1
J = 1
694 TR(I) = TR(I) + A(I,J)*SR(J)
SRI(I) = TR(I) + A(I,J)*SR(J)
J = J + 1
IF (J < 10) 693, 693, 694
696 I = I + 1
IF (I < 10) 692, 692, 695
698 I = I
698 CR(I) = -C(I)
SRI(I) = 0.
J = 1
698 CR(I) = CR(I) - Y1(I,J)*TR(J)
SRI(I) = SRI(I) + YR(I,J)*TR(J)
J = J + 1  
IF (J - IB) 698, 698, 700  
700 I = I + 1  
IF (I - IB) 696, 696, 702  
702 I = 1  
704 J = 1  
706 SR(I) = SR(I) + YR(I,J)*TI(J)  
SI(I) = SI(I) + YI(I,J)*TI(J)  
J = J + 1  
IF (J - IB) 706, 706, 708  
708 I = I + 1  
IF (I - IB) 704, 704, 710  
710 PUNCH 172  
PUNCH 183  
J = 1  
GO TO 656  
720 IF (W - WX) 722, 740, 740  
722 W = W * DW  
724 I = 1  
724 J = 1  
726 IF (YI(I,J)) 728, 732, 730  
728 YI(I,J) = YI(I,J)/DW  
GO TO 732  
730 YI(I,J) = YI(I,J)*DW  
732 J = J + 1  
734 I = I + 1  
IF (I - IB) 726, 726, 734  
734 I = I + 1  
IF (I - IB) 725, 725, 600  
740 IF (SENSE SWITCH 3) 742, 744  
742 PRINT 156  
PAUSE  
744 IF (SENSE SWITCH 1) 500, 746  
746 PRINT 153  
GO TO 400  
300 PRINT 180  
L = 1  
GO TO 415  
301 PRINT 161, I  
L = 1  
GO TO 464  
303 PRINT 165  
L = 1  
GO TO 464  
304 PRINT 165  
L = 1  
GO TO 481  
305 PRINT 173, I  
L = 1  
GO TO 706  
306 PRINT 154  
PUNCH 154  
GO TO 746  
150 FORMAT (3131)
151 FORMAT (3I3, E12.5)
152 FORMAT (3I3, E15.8)
153 FORMAT (9HNEXT CIRC)
154 FORMAT (20H INPUT ERROR, NO RUN)
155 FORMAT (18H CARDS OUT OF ORDER)
156 FORMAT (20H SS) ON TO TYPE INPUT
159 FORMAT (40H TYPE INPUT (FREQ), 552 ON TO CORRECT ERROR)
160 FORMAT (17H TYPE INPUT (PARAM))
161 FORMAT (2HI, I3, 3I) = 0
162 FORMAT (3E15.8)
164 FORMAT (6HFREQ =, E15.8)
165 FORMAT (// HFREQ =, E14.8, 8H RAD/SEC)
166 FORMAT (4HDW = 0)
163 FORMAT (1HE, I3, 2X, 2E16.8)
169 FORMAT (// 10X, 7HCIRCUIT, I3)
172 FORMAT (// 20H 1 VOLTAGE "!" = 0)
171 FORMAT (1HI, I3, 2X, 2E16.8)
172 FORMAT (// 26H 1 1-TH BRANCH CURRENT)
177 FORMAT (// 21H A-C CIRCUIT ANALYSIS)
178 FORMAT (8H B(, I3, 4H) N(, I3, 6H) TO N(* I3, 4H) =, 1E10.4, 5H L-C*, E10.4, 3H F=*, E10.4, 3H I *, E10.4)
179 FORMAT (8H DEP, B(* I3, 13H) CONTROLS B(* I3, 9H), DELTA =)
190 FORMAT (14HCIRCUIT OVERSIZED)
181 FORMAT (31H, 4E15.8)
182 FORMAT (12X, 5HVOLTS, 11X, 7H DEGREES)
183 FORMAT (12X, 4HAMPS, 12X, 7H DEGREES)
184 FORMAT (3I3, 4F12.5)
END
APPENDIX III

List of Symbols Used in Transient Analysis Program

DELTA is the integration time interval.
PI is the output time interval.
CUT is the termination time.
IB is the number of blocks.
NN is the number of nodes.
NV is the number of state variables.
NU is the number of state equation inputs.
LSR equals NC(1) + NR(1).
LLC equals LINK + NC(2).
LCG equals NC(2) + NR(2).
LC equals NN + NC(1).
LIB equals NN + NC(1) + NR(1).
LINK is the number of links, equals NC(1) + NR(1) + NL(1).
LC equals LINK + NC(2) + NR(2).
LR equals NR(1) + NR(2).
ZZ(J) is the block passive element matrix.
NC(2) is the number of capacitive tree branches.
NR(2) is the number of resistive tree branches.
NL(2) is the number of inductive tree branches.
NC(1) is the number of capacitive links.
NR(1)  is the number of resistive links.
NL(1)  is the number of inductive links.
LSN(J) gives the block number of the J-th entered source.
A(I, J) is the matrix inverted by the subroutine. Also used for the state variable A-matrix.
AA(I, J) is the vertex matrix.
AI(I, J) is \((AA_{12})^{-1}\).
CC(I)  is the block current source matrix.
EE(I)  is the block voltage source matrix.
DEP(I) is the dependent variable in the function generator.
XNDEP(I) is the independent variable in the function generator.
XMAG(I) is the magnitude quantity associated with sources.
FREQ(I) is the time quantity associated with sources.
TF(I)  is the clock for the sources.
Y(I)   is the value of the source generators.
IND(J) gives the type of generator associated with the J-th source.
QQ(I, J) is the fundamental cut-set matrix.
BB(I, J) is the fundamental loop matrix.
TRG(I, J), TIS(I, J), TVL(I, J) store intermediate results.
TT(I)  stores the source-defining characters.
INDX(I) is an index used in the matrix inversion routine.
I.L   stops the matrix inversion when it exceeds the size of the matrix.
LAGF is a flag used in matrix inversion. Also used in the input routine to indicate inductor, resistor, or capacitor.

LAGG is a flag used in the input to indicate link or tree branch.

I, J, K, L, M, N, are indices used in matrix operations; I, J, K also used as temporary storage in input.

R, RR, D, B are temporary storage used in input routine. Also used for temporary storage in the computations.

TEST is used to store the Alpha input to be checked.

LENG is the number of points in the function generator.

XT, XI, YT, TI, XTT are used in the interpolation connected with the function generator. YT ≥ y(t) ≥ YI; XT ≥ t ≥ XI;

\[ y(t) = YI + \frac{(YT - YI)(t - XI)}{(XT - XI)} \]

KEY points to the return address from the inversion subroutine.

B(I, J) is the state equation B-matrix.

AX(I, J), BX(I, J), TL(I, J) are used in forming the state equations.

\[ A(I, J) = AX(I, J) \times BX(I, J) \]
\[ B(I, J) = AX(I, J) \times TL(I, J). \]

TG(I, J), TR(I, J) correspond to \( S^{-1} \) and \( R^{-1} \) respectively.

X(I) is the value of the state variables at some time, t.

XX(I) is the predicted value of the state variable at time, \( t + \Delta t \).

DX(I) is the value of dx/dt at time, t.

E(I) is BB(I, J) * EE(J).
\( G(I) \) is \( Q(I, J) \times \overline{C}(J). \)

\( XE(I) \) is the node voltage.

\( XC(I) \) is the branch current.

\( U(I) \) is the state variable source vector.

\( TM(I) \) is temporary storage. Also contains the node-to-reference potential.

\( XS \) is the argument of the sine function.

\( T \) is time.

\( TP \) is the clock controlling the output.

\( KEZ \) points to the return address from the subroutine which determines \( u(t) \).

\( PDX \) is \( dx/dt \), first at \( t \), then at \( t + \Delta t \).
Transient Analysis

Start
Pass 1

Initialize

Read Block Data Card

Error

Form $AA_{i,j}$, $ZZ_i$, $EE_i$, $CC_i$

If $EE_i \neq 0$,

$n \geq 5$

ET

Read Source Specification Cards

112

Read Time Specifications

124

700

Error Termination

Pause
Transient Analysis

Initialize

\[ A_{i,j} = (AA_{12})^{-1} \]

Form \( B_{i,k} \)

Form \( TRG_{m,n} \)

\[
\begin{bmatrix}
1_{i,j} & F_{RGk,j} \\
-F_{RGi,j}/R_1 & 1/R_2
\end{bmatrix}
\]

Invert \( TRG_{m,n} \) and Put Back in \( TRG_{m,n} \)

Invert \( F'_{SCi,j} \) and Place it in \( TIS_{i,j} \)
Transient Analysis

970

Invert $F_{LT1,j}$ and Place in $TVL_{1,j}$

990

Call Pass 2

End Pass 1
Transient Analysis

Start
Pass 2

Initialize

Form
\[ C_{i,j} = C_{i,j} + F_{i} \cdot *C_{i,k} *F_{k} \]

Invert C

Form
\[ \mathcal{L}_{i,j} = L_{i1} + F_{i} \cdot *L_{i,k} *F_{k} \]

Invert \( \mathcal{L} \)

Form
\[ \mathcal{R}_{i,j} = R_{i1} + F_{i} \cdot *R_{i,k} *F_{k} \]

\[ AX = \begin{bmatrix} C^{-1} & 0 \\ 0 & \mathcal{L}^{-1} \end{bmatrix} \]
Invert $\mathcal{R}$

$TR_{i,j} \leftarrow \mathcal{R}^{-1}$

Form

$$\mathcal{S} = \frac{1}{R_{2i} + R_{Gi,k} \cdot \left(1/R_{1k}\right) \cdot R_{Gk,j}}$$

Invert $\mathcal{S}$

$TG_{i,j} \leftarrow \mathcal{S}^{-1}$

Form $TL_{i,j}$

$TL_{i,j}$ (Defined in Text) is the Second Term in the Product Forming $B_{i,j}$
Transient Analysis

BX (Defined in the Text) is the Second Term in the Product for $A_{i,j}$

Form $BX_{i,j}$

$A_{i,j} = AX_{i,k} * BX_{k,j}$

$B_{i,j} = AX_{i,k} * TL_{k,j}$

Call Pass 3

End Pass 2
Transient Analysis

Start
Pass 3

046

Initialize

$E_k \leftarrow Y_k \ast EE_k$

$C_k \leftarrow Y_k \ast CC_k$

047

056

$U1_k \leftarrow QQ_{k,j} \ast C_j$

$U2_k \leftarrow BB_{k,j} \ast E_j$

$U_k = \begin{bmatrix} U1_k \\ U2_k \end{bmatrix}$

072

time $\neq 0$

KEZ
time $= 0$

218

200
Transient Analysis

200

Initialize
KEZ = 2

203

\[ f_i(N) = A_{i,j} \cdot X_j + B_{i,j} \cdot U_k \]

PDX \leftarrow f_i

215

DX_i \leftarrow PDX

P(X_i) = X_i + \Delta t \cdot PDX

XX_i \leftarrow P(X_i)

218

i ≤ \text{number of state variables}

239

> i

f_i(N+1) = A_{i,j} \cdot XX_j + B_{i,j} \cdot U_k

PDX \leftarrow f_i(N+1)

226

C_i(N+1) = X_i + \frac{1}{2} (DX_i + PDX) \cdot \Delta t

X_i \leftarrow C_i(N+1)

i ≤ \text{number of state variables}

i > \text{number of state variables}
Transient Analysis

```
239

Output Desired

\[ t \]

No Output

\[ 470 \]

Compute and Output Node-to-Reference Potential and Block Current

\[ 470 \]

End Computation

\[ 216 \]

Increment \( t \)

\[ 217 \]

Find New Values of \( Y_i \)

Values of \( Y_i \)
Defined in Statements 300-354

\[ 047 \]

\[ 472 \]

Type Last Output if not Already Typed

Type "To Extend Time Push Start, Type New Final Time"

Pause

Accept New Final Time

\[ 470 \]
```
TRANSIENT ANALYSIS - LINEAR, PASSIVE CIRCUITS
NODE VOLATGES AND BRANCH CURRENTS
CARD INPUT/OUTPUT
OPTIONAL CONSOLE TYPEWRITER OUTPUT

**NOTE** A TREE CONTAINING THE MAXIMUM NUMBER OF
CAPACITORS AND THE MINIMUM NUMBER OF INDUCTORS
MUST BE FORMED IN THE CIRCUIT. THE CIRCUIT
BLOCKS MUST BE NUMBERED WITH CAPACITOR LINKS
FIRST, THE RESISTOR LINKS, INDUCTOR LINKS,
CAPACITOR TREE BRANCHES, RESISTOR TREE BRANCHES,
AND FINALLY INDUCTOR TREE BRANCHES.

* INPUT SPECIFICATION - PPQ FREE FORMAT

FIRST CARD - NO. OF BLOCKS - MAX. OF 9
NO. OF NODES (EXCLUDING REFERENCE NODES
WHICH ARE GIVEN THE NUMBER ZERO) - MAX. OF 8
THE CHARACTER ZERO

ONE CARD FOR EACH BLOCK, STARTING WITH BLOCK 1
CONTINUING IN SEQUENCE -
BLOCK NO.
SOURCE NODE NO.
SINK NODE NO.
BLOCK RESISTANCE (IN OHMS)
BLOCK CAPACITANCE (NEGATIVE NO.) IN
FARADS, OR BLOCK INDUCTANCE (POSITIVE
NO.) IN HENRIES
'VALUE' OF VOLTAGE SOURCE (IN VOLTS)
'VALUE' OF INDEPENDENT CURRENT SOURCE
(IN AMPS).

THE NEXT CARDS DESCRIBE THE NETWORK SOURCES. THE
SOURCES ARE LISTED IN ORDER OF THEIR BLOCK NO.
SHOULD A BLOCK CONTAIN BOTH VOLTAGE AND CURRENT
SOURCES, THE VOLTAGE SOURCE IS LISTED FIRST.
A MAX. OF FIVE SOURCES ARE ALLOWED.

**NOTE** THE MAGNITUDE OF SOURCES USED IN COMPUTATION
IS THE SOURCE VALUE (GIVEN ON THE BLOCK
SPECIFICATION CARD) TIMES THE MAGNITUDE OF THE
CORRESPONDING GENERATOR.

FIVE TYPES OF SOURCES ARE ALLOWED -
ONE CARD FOR EACH POINT OF THE DEFINED VALUE OF INDEPENDENT VARIABLE (TIME)
VALUE OF FUNCTION AT THAT TIME.

NOTE ** THE VALUE OF THE FUNCTION IS ASSUMED TO BE
ZERO AT TIME = 0 UNLESS OTHERWISE SPECIFIED.

PERIODIC FUNCTION GENERATOR.

SIMILAR TO THE NON-PERIODIC FUNCTION GENERATOR EXCEPT THAT THE FUNCTION ACTS CYCLICALLY.

WHEN THE LAST VALUE IN THE DEFINED TIME DOMAIN IS
REACHED AND EXCEEDED, THE FUNCTION STARTS OVER
AT ITS INITIAL VALUE. THE PERIODIC FUNCTION
GENERATOR IS SPECIFIED BY THE FOLLOWING CARDS—
FIRST CARD — THE 4 CHARACTERS FUNC
REMAINING CARDS SAME AS FOR NON-PERIODIC FUNCTION.

** NOTE ** ONLY ONE FUNCTION GENERATOR MAY BE DECLARED.

SINE WAVE GENERATOR

SPECIFIED BY THE FOLLOWING CARDS —
FIRST CARD — THE 4 CHARACTERS SINE
SECOND CARD — MAGNITUDE (ZERO TO PEAK) OF GENERATOR
PERIOD (IN SECONDS)

SQUARE WAVE GENERATOR

SPECIFIED BY THE FOLLOWING CARDS —
FIRST CARD — THE 4 CHARACTERS SQUA
SECOND CARD — MAGNITUDE (ZERO TO PEAK) OF GENERATOR
PERIOD (IN SECONDS)

STARTING AT TIME = 0., THIS GENERATOR HAS A VALUE
OF + (MAGNITUDE) UNTIL TIME = (PERIOD)/2,
WHEN IT ASSUMES THE VALUE OF -(MAGNITUDE), ETC.

STEP FUNCTION GENERATOR

SPECIFIED BY THE FOLLOWING CARDS —
FIRST CARD — THE 4 CHARACTERS STEP
SECOND CARD — MAGNITUDE OF GENERATOR
TIME DELAY (IN SECONDS)

STARTING AT TIME = (DELAY), THE GENERATOR HAS A
VALUE = (MAGNITUDE). PRIOR TO THIS TIME, THE
GENERATOR HAS VALUE = 0.

NON-PERIODIC FUNCTION GENERATOR.

A MAX. OF 20 POINTS CAN BE SPECIFIED. LINEAR
INTERPOLATION IS USED BETWEEN POINTS. THE FINAL
VALUE IS MAINTAINED IN THE DEFINED TIME DOMAIN
IN EXCEEDED, THE NON-PERIODIC FUNCTION GENERATOR
IS SPECIFIED BY THE FOLLOWING CARDS—
FIRST CARD — THE 4 CHARACTERS FUNC
SECOND CARD — THE NUMBER OF POINTS TO BE DEFINED —
MAX. OF 20.
**NOTE**

The only allowable character sets which can be used to define a generator are: FUNC, FUND, 
SINE, SQUARE, or STEP. The use of any other characters will cause the machine to go into 
MANUAL mode. This will be interpreted as a zero input. If this is correct press the start 
button and the program will continue. If this is not correct, sense switch 2 should 
be turned on before pushing start. The program will return to read the corrected card.

**LAST CARD** - THE INTEGRATION INTERVAL (IN SEC) 
THE OUTPUT INTERVAL (IN SEC) 
THE FINAL TIME (IN SEC)

**EXAMPLE OF FORM** ---

```
5 2 0
1 0 1 100. 0. 1. 0.
2 1 2 0. 5.6e-3 0. 0.
3 2 0 0. 2.6e-4 0. 0.
4 0 2 0. -5.6e-6 0. 0.
5 2 1 1000. 0. 0. 0.5
```

SINE 
STEP 
0. 0.

**OPERATION ---**

**NOTE** Program must be reloaded for each circuit to 
be analysed. The program is in 3 passes with 
pass 2 and 3 loaded automatically. 
Data is placed between pass 1 and pass 2.
COMMON DELTA, PI, CUT
COMMON IQ, NN, NV, NU
COMMON LSR, LL, LC, LNC, L18, LINK, LC, LR
DIMENSION ZZ(9), NC(2), NL(2)
DIMENSION CN(5), NX(2), A(9,9), AA(9,9)
DIMENSION AI(9,9)
DIMENSION CC(9), LE(9)
DIMENSION DEP(20), XNDEP(20)
DIMENSION XHAG(5), FREQ(5)
DIMENSION TI(5)
DIMENSION Y(5), IND(5)
DIMENSION QQ(9,9)
DIMENSION BB(9,9)
DIMENSION TRG(9,9)
DIMENSION TIS(4,4), TVL(4,4)

DIMENSION TT(5)
DIMENSION INDEX(9)

PRINT 510
PUNCH 509
TT(1) = .645543E-4
TT(2) = .645557E-4
TT(3) = .495545E12
TT(4) = .586641E12
TT(5) = .639557E12
GO TO 401

MATRICES INVERSE

200 K = 1
LAGF = 0
IF (N) 202, 300, 202
202 INDEX(K) = 0
IF (K - N) 202, 202, 204
204 K = 1
LL = 0
206 IF (A(K,K)) 208, 260, 208
208 A(K,K) = 1./A(K,K)
LL = LL + 1
INDEX(K) = 1
I = 1
210 IF (I - K) 212, 220, 212
212 J = 1
214 IF (J - K) 216, 216, 216
216 A(I,J) = A(I,J) - A(I,K)*A(K,J)*A(K,K)
218 J = J + 1
IF (J - N) 214, 214, 220
220 I = I + 1
IF (I - N) 210, 210, 230
230 J = 1
232 IF (J - K) # 244, 246, 236
244 A(J,K) = 0
246 A(J, K) = A(J,K)
248 J = J + 1
249 IF (J = N) 232, 242, 240
240 IF (LAGF) 250, 248, 250
242 J = J + 1
243 IF (K - N) 206, 206, 300
250 K = 1
252 IF (J # K) 254, 256, 254
254 J = J + 1
256 IF (K = N) 252, 252, 270
270 IF (J = N) 710, 300, 300
258 IF (A(K,K) = 208, 254, 208
260 IF (LAGF) 250, 262, 250
262 K = K + 1
266 LAGF = 1
267 IF (K = N) 206, 206, 710
300 GO TO (301, 970, 360, 600, 640, 680, 960, 980), KEY

C
700 PRINT 701, I
702 GO TO 706
704 PRINT 705
710 PRINT 711, KEY
706 PRINT 707

C
401 LAGF = -1
402 NC(i) = 0
404 NL(1) = 0
406 NL(i) = 0
408 NL(J) = 0
410 HI(1) = 0
412 N = 0
414 READ 500, IB, NN, IC
416 LINK = IB - NN
418 J = 1
420 K = 1
422 AA(J,K) = 0
424 A(J,K) = 0
426 BR(J,K) = 0
428 K = K + 1
430 IF (K # IB) 408, 408, 410
432 J = J + 1
434 IF (J # IB) 406, 406, 400
440 READ 501, I, J, K, R, RR, D, B
PUNCH 514, 1, J, 7, 2, PR, 0, B
IF (I - LINK) 414, 214, 266
412 IF ((LAGG - 1) 404, 304, 414
414 LAGG - 2
1AGI = -1
414 IF (LAGI) 416, 424, 430
416 IF (GR) 418, 422, 422
418 IF (PR) = PR
    IF (I) 100, 420, 100
420 NL (LAGG) = NL (LAGG) + 1
    GO TO 440
427 LAGI = 0
424 IF (R) 476, 428, 426
426 /AGI = R
    NL (LAGG) = NL (LAGG) + 1
    IF (RPR) 700, 440, 700
428 LAGI = 1
430 IF (RPR) 700, 700, 432
432 IF (I) 442, 442, 442
442 AA (J, 1) = 1
444 IF (R) 446, 448, 446
446 AA (K, 1) = -1
448 IF (I) = 0
    IF (D) 449, 450, 449
449 NU = NU + 1
    LSN (NU) = 1
    IF (NU - 5) 450, 450, 704
450 CC (I) = B
    IF (B) 452, 454, 452
452 IF (LAGF) 700, 453, 700
453 NU = NU + 1
    LSN (NU) = -1
    IF (NU - 5) 454, 454, 704
454 N = N + 1
    IF (N = 18) 400, 400, 456
456 IF (NC (1) = NC (2)) 455, 455, 702
455 IF (NL (2) = NL (1)) 112, 112, 702
C
122 I = 0
122 IF (I - NU) 121, 124, 124
121 I = 1 + I
133 PEAAD 505, TEST
    IF (LSN (I)) 111, 700, 115
111 K = -LSN (I)
    PUNCH 514, * K
    GO TO 110
115 PUNCH 514, LSN (I)
110 Y (I) = 0
    TF (I) = 0
    J = 0
114  J = J + 1
115  IF (J = 5) 116, 116, 160
116  IF (115.1 = 11(J)) 1 160, 120, 118
118  GO TO 114
120  IND(J) = J
121  GO TO (130, 140, 150, 170, 160), J

C
C DEFINE FUNCT GEN (N/N-PERIODIC)
130  PRINT 556
131  PUNCH 557
136  READ 503, LENG
137  J = 1
138  LFINR = 2
139  READ 503, XMAG(I), FREQ(I)
140  IF (XMAG(I)) 133, 131, 133
133  XNDEP(I) = 0.
134  DEP(I) = 0.
135  LENG = LENG + 1
136  J = 2
137  XNDEP(J) = XMAG(I)
138  DEP(J) = FREQ(I)
139  Y(I) = DEP(I)
140  J = J + 1
141  READ 503, XNDEP(J), DEP(J)
142  J = J + 1
143  IF (J = LENG) 132, 132, 137
137  XI = XNDEP(2)
138  XI = 0.
139  YI = DEP(1)
140  XIT = (DEP(2) - DEP(1)) / XI
141  J = 1
142  PUNCH 516, XNDEP(J), DEP(J)
143  J = J + 1
144  IF (J = LENG) 138, 138, 122

C
C DEFINE FUNCT GEN (PERIODIC)
140  PRINT 558
141  PUNCH 559
142  GO TO 136

C
C DEFINE SINE INPUT
150  PRINT 560
151  READ 503, XMAG(I), FREQ(I)
152  PUNCH 561 XMAG(I), FREQ(I)
153  FREQ(I) = 6.2831853/FREQ(I)
154  GO TO 172

C
C DEFINE STEP INPUT
160  PRINT 562
161  READ 503, XMAG(I), FREQ(I)
162  PUNCH 563 XMAG(I), FREQ(I)
163  IF (FREQ(I)) 164, 164, 122
164  Y(I) = XMAG(I)
IN0(1) - 6
GO TO 122

C DEFINE SQUARE WAVE
170 PRINT 564
READ 503, XMA(1), FREQ(1)
PUNCH 565, XMA(1), FREQ(1)
FREQ(1) = 0.8*FREQ(1)
Y(i) = XMA(i)
GO TO 122

C 180 PRINT 541
PAUSE
IF (SENSE SWITCH 2) I13, 181
181 PUNCH 542
IND(I) = 6
GO TO 122

C 124 READ 507* DELTA, PI, CUT
PUNCH 566* DELTA
PUNCH 567* PI
PUNCH 568* CUT

C 457 N = NN
NV = NC(2) + NL(1)
LC = IB - NL(2)
LSR = NC(1) + NR(1)
LLC = LINK + NC(2)
LCG = NC(2) + NR(2)
LR = NR(1) + NR(2)
LNC = NN + NC(1)
LIB = IB - NL(1)

C KEY = 1
K = LINK + 1
L = 1
458 J = 1
460 A(J,K) = AA(J,K)
J = J + 1
IF (J = NN) 460, 460, 462
462 K = K + 1
I = I + 1
IF (K = IB) 458, 458, 200

C 394 I = 1
302 J = 1
M = LINK + 1
RR(K*K) = I
304 I = 1
QQ(J*K) = 0.
306 QQ(J,K) = QQ(J,K) + A(J*1)*AA(1*K)
i = I + 1
C
900 K = 0
KEY = 2
901 IF (K = NR(1)) 902, 903, 903
902 K = K + 1
A(K,K) = 1.
GO TO 901
903 J = INK + NC(2)
904 IF (K = LR) 906, 906, 908
906 K = K + 1
J = J + 1
A(K,J) = 1/ZZ(J)
GO TO 904
908 N = NR(1)
K = NC(2)
910 IF (N = LR) 912, 200, 200
912 N = N + 1
K = K + 1
J = 0
M = NC(1)
914 IF (J = NR(2)) 916, 910, 910
916 J = J + 1
M = M + 1
A(J,N) = -QQ(K,M)
A(N,J) = QQ(K,M)/ZZ(M)
GO TO 914
C
920 r = 0
922 IF (r = LR) 924, 950, 950
924 K = K + 1
J = 0
926 IF (J = LR) 928, 922, 922
928 J = J + 1
TRG(J,K) = A(J,K)
A(J,K) = 0.
GO TO 926
C
950 N = 0
KEY = 7
952 IF (N - NC(1)) 954, 200, 200
954 N = N + 1
J = 0
956 IF (J - NC(1)) 958, 952, 952
958 J = J + 1
A(J,N) = -QQ(J,N)
GO TO 956

C
960 K = 0
962 IF (K - NC(1)) 964, 970, 970
964 K = K + 1
J = 0
966 IF (J - NC(1)) 968, 962, 962
968 J = J + 1
TIS(J,K) = A(J,K)
A(J,K) = 0.
GO TO 966

C
970 N = 0
KEY = 8
K = LCG
972 IF (N - NL(2)) 974, 200, 200
974 N = N + 1
K = K + 1
J = 0
M = LSR
976 IF (J - NL(2)) 978, 972, 972
978 J = J + 1
M = M + 1
A(J,N) = -QQ(K,M)
GO TO 976

C
980 K = 0
982 IF (K - NL(2)) 984, 990, 990
984 K = K + 1
J = 0
986 IF (J - NL(2)) 988, 982, 982
988 J = J + 1
TVI(J,K) = A(J,K)
A(J,K) = 0.
GO TO 986
990 CONTINUE

C
PAUSE
PAUSE

C
500 FORMAT (3E3)
501 FORMAT (3E3, 4E15.6)
502 FORMAT (3E15.6)
503 FORMAT (2E15.6)
504 FORMAT (13)
505 FORMAT (A4)
FORMAT (5HERROR, 13)
END
C

DO l00 K = 1, 100
C
C
C
C

C

C

C

DIMENSION A(9,9)
DIMENSION B(9,9)
DIMENSION IL(9,9)
DIMENSION T6(9,9)
DIMENSION TR(9,9)
DIMENSION IN(X(9,9)

PRINT 540
540 FORMAT (25HTRANSIENT ANALYSIS PASS 2)
010 K = 1
012 J = 1
014 A(I,J,K) = 0.
B(I,J,K) = 0.
J = J + 1
IF (J = NV) 014, 014, 016
016 K = K + 1
IF (K = 18) 012, 012, 335
C
C
C

MATRIX INVERSE

C

200 K = 1
IF (K = 1) 202, 300, 202
202 INX(K) = 0
K = K + 1
IF (K = N) 202, 202, 204
204 K = 1
IL = 0
206 IF (A(K,K) = 208, 260, 208
208 A(K,K) = 1./A(K,K)
LL = LL + 1
INX(K) = 1
I = 1
210 IF (I - K) = 212, 220, 212
212 I = 1
214 IF (J - K) = 216, 214, 216
216 A(I, J) = A(I, J) - A(I, K) * A(K, J) * A(K, K)
218 J = J + 1
   IF (J = N) = 214, 214, 220
220 I = I + 1
   IF (I = N) = 210, 210, 230
230 J = 1
232 IF (J - K) = 234, 236, 236
234 A(K, J) = 0 * - A(K, J) * A(K, K)
   A(J, K) = A(J, K) * A(K, K)
236 J = J + 1
   IF (J = N) = 232, 232, 240
240 IF (LAGF) = 250, 242, 250
242 K = K + 1
   IF (K = N) = 206, 206, 300
250 K = 1
252 IF (INDEX(K)) = 254, 256, 254
254 K = K + 1
   IF (K = N) = 252, 252, 270
270 IF (LJ - N) = 700, 300, 300
272 IF (A(K, K)) = 208, 254, 208
276 IF (LAGF) = 250, 262, 250
262 K = K + 1
   LAGF = 1
   IF (K = N) = 206, 206, 700
300 GO TO (301, 920, 360, 640, 680, 960, 980), KEY

335 J = 0
   M = LINK
336 IF (J = NC(2)) = 337, 340, 340
337 J = J + 1
   M = M + 1
   A(J, J) = ZZ(M)
   GO TO 336
340 N = 0
   KEY = 3
344 IF (N = NC(2)) = 344, 200, 200
344 N = N + 1
   J = 0
346 IF (J = NC(1)) = 348, 342, 342
348 J = J + 1
   D = ZZ(J) * QQ(IN, J)
   I = 0
350 IF (I = NC(2)) = 352, 346, 346
352 I = I + 1
   A(I, IN) = A(I, IN) + D * QQ(IN, J)
   GO TO 350
360 J = 0
361 IF (J = NC(2)) = 362, 370, 370
362 J = J + 1
   I = 1
363 AX(I,J) = A(I,J)
   A(I,J) = 0
367 IF (I = NC(2)) 363, 363, 361
370 J = 0
   M = LSR
372 IF (J = NL(1)) 374, 380, 380
374 J = J + 1
   M = M + 1
   A(J,J) = ZZ(M)
   GO TO 372
380 N = 0
   L = LSR
   KEY = 4
382 IF (L = LINK) 384, 200, 200
384 N = N + 1
   L = L + 1
   J = LC
386 M = NC(2) + NR(2)
388 J = J + 1
390 IF (K = LINK) 392, 386, 386
392 J = J + 1
   K = K + 1
   A(I,N) = A(I,N) + D*QG(M,L)
   i = 0
   K = LSR
600 J = 0
601 IF (M = NV) 602, 610, 610
602 J = J + 1
603 AX(N,M) = A(I,J)
   A(I,J) = 0
   I = I + 1
   N = N + 1
   IF (N = NV) 603, 603, 601
610 J = 0
612 IF (J = NR(1)) 613, 620, 620
613 J = J + 1
   M = M + 1
   A(J,J) = ZZ(M)
   GO TO 612
620 N = 0
   L = NC(1)
KEY = 5
627 IF (N = NR(1) ) 624, 200, 200
624 N = N + 1
I = I + 1
J = LLC
M = NC(2)
626 IF (J = LC) 628, 622, 622
628 J = J + 1
M = M + 1
R = ZZ(J)*QQ(M,L)
I = 0
K = NC(1)
630 IF (I = NR(1) ) 632, 626, 626
632 I = I + 1
K = K + 1
A(I,N) = A(I,N) + D*QQ(M,K)
GO TO 630
640 J = 0
641 IF (J = NR(1) ) 642, 650, 650
642 J = J + 1
I = 1
643 TR(I,J) = A(I,J)
A(I,J) = 0.
I = I + 1
IF (I = NR(1) ) 643, 643, 641
650 J = 0
M = LLC
652 IF (J = NR(2) ) 654, 660, 660
654 J = J + 1
M = M + 1
A(J,J) = L/ZZ(M)
GO TO 652
660 N = 0
L = NC(2)
KEY = 6
662 IF (N = NR(2) ) 664, 200, 200
664 N = N + 1
L = L + 1
J = NC(1)
666 IF (J = LINK + NL(1) ) 668, 662, 662
668 J = J + 1
D = QQ(L,J)/ZZ(J)
I = 0
K = NC(2)
670 IF (I = NR(2) ) 672, 666, 666
672 I = I + 1
K = K + 1
A(I,N) = A(I,N) + D*QQ(K,J)
GO TO 670
680 J = 0
682 IF (J = NR(2) ) 683, 800, 800
683 : = J + 1
I = 1
684 \[ A(i, j) = 0 \]
684 \[ i = 1 + 1 \]
684 IF (i \neg NR(1) ) \[ 684, 684, 682 \]

C

800 \( K = 0 \)
802 \( N = LNC \)
804 IF (K \neg NR(1) ) \[ 806, 812, 812 \]
806 \( K = K + 1 \)
806 \( N = N + 1 \)
807 \( J = 0 \)
807 IF (J \neg NC(2) ) \[ 808, 805, 805 \]
808 \( J = J + 1 \)
809 \( L = NC(1) \)
810 IF (i \neg NR(1) ) \[ 810, 807, 807 \]
810 \( i = 1 + 1 \)
812 \( K = 0 \)
814 \( L = NC(2) \)
816 IF (K \neg NR(2) ) \[ 816, 822, 822 \]
816 \( K = K + 1 \)
816 \( L = L + 1 \)
818 IF (J \neg NR(1) ) \[ 820, 814, 814 \]
820 \( J = J + 1 \)
822 \( K = 0 \)
824 IF (K \neg NC(2) ) \[ 826, 832, 832 \]
826 \( K = K + 1 \)
826 TL(K,K) = 1
827 IF (J \neg NR(2) ) \[ 828, 824, 824 \]
828 \( J = J + 1 \)
829 IF (i \neg NR(1) ) \[ 830, 827, 827 \]
830 \( i = 1 + 1 \)
830 L = L + 1
\[ TL(K,M) = TL(K,M) + TL(K,L) \times A(I,J) \]

GO TO 870

C

832 K = 0
N = NC(2)
834 IF (K = NR(2)) 836, 850, 850
836 K = K + 1
N = N + 1
J = NC(2)
M = ISR
838 IF (M = LINK) 840, 834, 844
840 M = M + 1
J = J + 1
I = 0
L = NC(2)
842 IF (I = NR(2)) 844, 838, 838
844 I = I + 1
L = L + 1
TL(J,N) = TL(J,N) + QQ(L,M) \times TG(I,K)
GO TO 841

C

850 K = 0
N = NC(1)
851 IF (K = NR(1)) 855, 860, 860
855 K = K + 1
N = N + 1
J = 0
M = NC(2)
858 IF (J = NR(2)) 856, 851, 891
858 J = J + 1
M = M + 1
A(J,K) = -QQ(M,N)/ZZ(N)
GO TO 856

C

860 K = NC(2)
N = LIB
862 IF (K = NV) 864, 400, 400
864 K = K + 1
N = N + 1
TL(K,N) = 1.
J = 0
M = LNC
866 IF (J = NR(1)) 868, 862, 862
868 M = M + 1
J = J + 1
I = 0
L = NC(2)
870 IF (I = NR(2)) 872, 866, 866
872 I = I + 1
L = L + 1
TL(K,M) = TL(K,M) + TL(K,L) \times A(I,J)
GO TO 870

C
400 \( \text{K} = \text{LC} \)
\( \text{N} = \text{LCU} \)
402 IF \((\text{K} = \text{M})) \) \( \text{GO TO 404} \)
404 \( \text{K} = \text{K} + 1 \)
\( \text{N} = \text{N} + 1 \)
\( \text{J} = \text{NC(2)} \)
\( \text{M} = \text{LSR} \)
406 IF \((\text{J} = \text{NV})) \) \( \text{GO TO 408} \)
408 \( \text{J} = \text{J} + 1 \)
\( \text{M} = \text{M} + 1 \)
\( iL(J,N) = \text{QQ(N,M)}*ZZ(K) \)
\( \text{GO TO 406} \)
410 \( \text{K} = 0 \)
\( \text{N} = \text{NN} \)
412 IF \((\text{K} = \text{NC(1)})) \) \( \text{GO TO 414} \)
414 \( \text{K} = \text{K} + 1 \)
\( \text{N} = \text{N} + 1 \)
\( \text{J} = 0 \)
416 IF \((\text{J} = \text{NC(2)})) \) \( \text{GO TO 418} \)
418 \( \text{J} = \text{J} + 1 \)
\( TL(J,N) = -\text{QQ(J,K)}*ZZ(K) \)
\( \text{GO TO 416} \)

C
420 CONTINUE
880 \( \text{K} = 0 \)
882 IF \((\text{K} = \text{NC(2)})) \) \( \text{GO TO 884} \)
884 \( \text{K} = \text{K} + 1 \)
\( \text{J} = 0 \)
886 IF \((\text{J} = \text{NC(2)})) \) \( \text{GO TO 888} \)
888 \( \text{J} = \text{J} + 1 \)
\( BX(J,K) = 0 \)
\( I = \text{LNC} \)
\( L = \text{NC(1)} \)
890 IF \((\text{L} = \text{LSR})) \) \( \text{GO TO 892} \)
892 \( I = I + 1 \)
\( L = L + 1 \)
\( BX(J,K) = BX(J,K) + TL(J,I)*QQ(K,L) \)
\( \text{GO TO 890} \)

C
893 \( \text{K} = \text{NC(2)} \)
\( \text{N} = \text{LSR} \)
895 IF \((\text{K} = \text{NV})) \) \( \text{GO TO 897} \)
897 \( \text{K} = \text{K} + 1 \)
\( \text{N} = \text{N} + 1 \)
\( \text{J} = \text{NC(2)} \)
899 \( \text{L} = \text{L} + 1 \)
\( BX(J,K) = BX(J,K) - TL(J,L)*QQ(L,N) \)
\( \text{GO TO 899} \)
C

760 K = NC(2)
N = LSR
762 IF (K = NV) 763, 900, 900
763 K = K + 1
N = N + 1
J = 0
764 IF (J = NC(2)) 765, 762, 762
765 J = J + 1
BX(J,K) = -QQ(J,N)
L = NC(2)
766 IF (L = LCG) 767, 768, 768
767 L = L + 1
BX(J,K) = BX(J,K) + TL(J,L) * QQ(L,N)
GO TO 766
768 BX(K,J) = -BX(J,K)
GO TO 764

C

900 K = 0
901 IF (K = NV) 902, 910, 910
902 K = K + 1
J = 0
903 IF (J = NV) 904, 901, 901
904 J = J + 1
A(J,K) = 0
I = 0
905 IF (I = NV) 906, 903, 903
906 I = I + 1
A(J,K) = A(J,K) + AX(J,I) * BX(I,K)
GO TO 905

C

910 K = 0
911 IF (K = 1H) 912, 920, 920
912 K = K + 1
J = 0
913 IF (J = NV) 914, 911, 911
914 J = J + 1
B(J,K) = 0
I = 0
915 IF (I = NV) 916, 913, 913
916 I = I + 1
B(J,K) = B(J,K) + AX(J,I) * TL(I,K)
GO TO 915

C

920 CONTINUE
PAUSE
PAUSE
700 PRINT 701, KEY
701 FORMAT (SHERROR, 13)
END

C
STATE VARIABLES

COMMON DELTA, PI, CUT
COMMON IN, NV, NU
COMMON IER, IE, IC, LCO, LNC, LI0, LINK, LC, LR

DIMENSION Z2(9), NC(2), NL(2)
DIMENSION LSN(9), NR(2), A1(9,9), AA(9,9)
DIMENSION A1(9,9)
DIMENSION CC(9), IC(9)
DIMENSION XDP(20), XNDP(20)
DIMENSION XMA(15), FREQ(5)
DIMENSION IF(5)
DIMENSION Y(9), IND(5)
DIMENSION QQ(9,9)
DIMENSION BB(9,9)
DIMENSION TRG(9,9)
DIMENSION TIS(4,4), TVL(4,4)

DIMENSION B(9,9)

DIMENSION X(9), XX(9)
DIMENSION DX(9)
DIMENSION E(9), C(9)
DIMENSION XE(9), XC(9)
DIMENSION U(9)
DIMENSION TM(9)

PRINT 540

KEZ = 1
J = 1
046 X(J) = 0*
J = J + 1
IF (J - NV) 046, 046, 045
045 PUNCH 502
PUNCH 503

FIND U(J)

047 CONTINUE
J = 1
048 C(J) = 0*
C(J) = 0*
J = J + 1
IF (J - IB) 048, 048, 050
050 J = 0
051 IF (J - NU) 052, 056, 056
052 J = J + 1
IF (LSN(J)) 055, 700, 054
054 K = LSN(J)
E(K) = Y(J)*EE(K)
GO TO 051
055 \( K = - \text{LSN}(J) \)
\( C(K) = Y(J) \times C(J) \)
GO TO 051

C
056 \( K = 1 \)
058 \( J = 1 \)
\( U(K) = 0 \)
060 \( U(K) = U(K) + QW(K, J) \times C(J) \)
\( J = J + 1 \)
IF \( (J = IB) \) 060, 060, 062
062 \( K = K + 1 \)
IF \( (K = N) \) 058, 058, 064
064 \( N = 0 \)
066 \( N = N + 1 \)
\( J = 1 \)
\( U(K) = 0 \)
068 \( U(K) = U(K) + B0(N, J) \times C(J) \)
\( J = J + 1 \)
IF \( (J = IB) \) 066, 066, 070
070 \( K = K + 1 \)
IF \( (K = IB) \) 066, 066, 072
072 GO TO (300, 216), KEZ

C
EVALUATE FUNCTIONS
300 \( TF(J) = TF(J) + \text{DELTA} \)
301 IF \( (TF(J) = XT) \) 312, 312, 302
302 NUPR = NUPR + 1
IF \( (NUPR = LFENG) \) 305, 305, 304
304 \( L = \text{IND}(J) \)
GO TO (322, 320), L
305 IF \( (TF(J) = XNDEP(NUPR)) \) 306, 306, 302
306 \( L = NUPR - 1 \)
\( XT = XNDEP(NUPR) \)
\( XI = XNDEP(2) \)
\( YT = DCP(NUPR) \)
\( YI = DCP(L) \)
\( XTT = (YT - YI)/(XT - XI) \)
312 \( Y(J) = YI + (TF(J) - XI) \times XTT \)
GO TO 217
320 \( TF(J) = TF(J) - XNDEP(LENG) \)
NUPR = 2
\( XI = XNDEP(2) \)
\( YI = DCP(1) \)
\( XTT = (DCP(2) - YI)/(XT - XNDEP(1)) \)
GO TO 301
322 \( Y(J) = DCP(LENG) \)
IND(J) = 6
GO TO 217
330 \( XS = T \times FRQ(J) \)
\( Y(J) = XMAG(J) \times \text{SIN}(XS) \)
GO TO 217

340 IF (J) = IF (J) + DELTA
IF (TF (J) - TREQ (J)) 217, 344, 344

344 Y(J) = XMAG(J)
IND(J) = 6
GO TO 217

350 IF (J) = IF (J) + DELTA
IF (TF (J) - TREQ (J)) 217, 352, 352

352 IF (J) = IF (J) - TREQ (J)
XMAG(J) = -XMAG(J)

354 Y(J) = XMAG(J)
GO TO 217

700 PRINT 571
571 FORMAT (5HER, ERROR)
GO TO 701

C
C
C FIND NODE VOLTS, BLOCK CURRENT
C

219 IF (T) 240, 238, 240
240 IF (IP = PI) 470, 246, 246
246 IP = TP - PI

238 PUNCH 501, T
IF (SENSE SWITCH 3) 235, 237

235 PRINT 501, T

237 K = 0
J = MIN

241 IF (K - NC(2)) 242, 243, 243
242 K = K + 1
J = J + 1
X(J) = X(K)
XC(J) = /Z(J)*OX(K)
GO TO 241

243 J = LSR

233 IF (K - NV) 244, 245, 245
244 K = K + 1
J = J + 1
XC(J) = X(K)
XE(J) = /Z(J)*OX(K)
GO TO 233

C

245 K = 0

236 IF (K - NC(1)) 247, 250, 250
247 K = K + 1
XE(K) = U(K)
J = 0

248 IF (J - NC(2)) 249, 236, 236
249 J = J + 1
XE(K) = XE(K) + QW(J,K)*X(J)
GO TO 248

C

250 K = LC
\( \text{\textbf{C}} \)

\( N = 1 \text{CG} \)

251 IF \((K = \text{IV})\) 252, 260, 260

252 \( K = K + 1 \)
\( i = i + 1 \)
\( \text{XC}(K) = 0(K) \)
\( J = \text{NC}(2) \)
\( M = \text{LSR} \)

253 IF \((J = \text{NV})\) 254, 251, 251

254 \( J = J + 1 \)
\( M = M + 1 \)
\( \text{XC}(K) = \text{XC}(K) - \text{QQ}(N,M) \times X(J) \)
GO TO 253

\( \text{\textbf{C}} \)

260 \( K = 0 \)
\( N = \text{NC} \)
\( i = \text{NC}(1) \)

261 IF \((K = \text{NR}(1))\) 262, 265, 265

262 \( K = K + 1 \)
\( N = N + 1 \)
\( L = L + 1 \)
\( \text{TM}(K) = U(N) \)
\( J = 0 \)

263 IF \((J = \text{NC}(2))\) 264, 261, 261

264 \( J = J + 1 \)
\( \text{TM}(K) = \text{TM}(K) + \text{QQ}(J,L) \times X(J) \)
GO TO 263

\( \text{\textbf{C}} \)

265 \( N = \text{NC}(2) \)

266 IF \((K = \text{LR})\) 267, 270, 270

267 \( K = K + 1 \)
\( N = N + 1 \)
\( \text{TM}(K) = U(N) \)
\( J = \text{NC}(2) \)
\( M = \text{LSR} \)

268 IF \((J = \text{NV})\) 269, 266, 266

269 \( J = J + 1 \)
\( M = M + 1 \)
\( \text{TM}(K) = \text{TM}(K) - \text{QQ}(N,M) \times X(J) \)
GO TO 268

\( \text{\textbf{C}} \)

270 \( K = 0 \)
\( L = \text{SR}(1) \)
\( N = \text{NC}(1) \)

271 IF \((K = L)\) 272, 276, 276

272 \( K = K + 1 \)
\( N = N + 1 \)
\( \text{XE}(N) = 0 \)
\( J = 0 \)

273 IF \((J = \text{LR})\) 274, 275, 275

274 \( J = J + 1 \)
\( \text{XE}(N) = \text{XE}(N) + \text{TRG}(K,J) \times \text{TM}(J) \)
GO TO 273

275 \( \text{XC}(N) = \text{XE}(N) / \text{ZZ}(N) \)
GO TO 271
270 IF (K = LR) 277, 260, 280
271 i = LR
N = LLC
GO TO 271
C
280 K = 0
L = LINK
281 IF (K = NC(1)) 282, 266, 266
282 K = K + 1
L = L + 1
IM(K) = XC(L) - U(K)
M = NC(1)
283 IF (M = LINK) 284, 281, 281
284 M = M + 1
IM(K) = IM(K) + QQ(K,M) * XC(M)
GO TO 283
C
286 K = 0
287 IF (K = NC(1)) 288, 440, 440
288 K = K + 1
XC(K) = 0
J = 0
289 IF (J = NC(1)) 291, 287, 287
291 J = J + 1
XC(K) = XC(K) + TIS(K,J) * TM(J)
GO TO 289
C
440 K = 0
L = LSR
N = LIB
441 IF (K = NL(2)) 442, 445, 445
442 K = K + 1
L = L + 1
N = N + 1
IM(K) = UI(N) - XE(L)
J = 0
M = LINK
443 IF (J = LCG) 444, 441, 441
444 J = J + 1
M = M + 1
IM(K) = IM(K) + QQ(J,L) * XE(M)
GO TO 443
C
445 K = 0
N = LC
446 IF (K = NL(2)) 447, 450, 450
447 K = K + 1
N = N + 1
XE(N) = 0
J = 0
448 IF (J = NL(2)) 449, 446, 446
449 J = J + 1
\text{XE}(N) = \text{XE}(N) + \text{TVL}(K, J) \times \text{TM}(J)

\text{GO TO 446}

\text{C}

450 \ K = 0
451 \text{IF } (K - \text{IB}) \ 452, 454, 454
452 \ K = K + 1
\text{XE}(K) = \text{XE}(K) - \text{E}(K)
\text{XC}(K) = \text{XC}(K) - \text{C}(K)
\text{GO TO 451}

\text{C}

454 \ K = 1
457 \ J = 1
L = \text{LINK} + 1
\text{TM}(K) = 0
458 \text{TM}(K) = \text{TM}(K) + \text{AI}(K, J) \times \text{XE}(L)
J = J + 1
L = L + 1
\text{IF } (J - \text{NN}) \ 458, 458, 456
456 \ K = K + 1
\text{IF } (K - \text{NN}) \ 457, 457, 460

\text{C}

460 \ K = 0
461 \text{IF } (K - \text{NN}) \ 462, 465, 465
462 \ K = K + 1
\text{PUNCH 504, } K, \text{ TM}(K)
\text{IF } (\text{SENSE SWITCH 3}) \ 463, 461
463 \text{PRINT 504, } K, \text{ TM}(K)
\text{GO TO 461}

\text{C}

465 \ K = 0
466 \text{IF } (K - \text{IB}) \ 467, 470, 470
467 \ K = K + 1
\text{PUNCH 508, } K, \text{ XC}(K)
\text{IF } (\text{SENSE SWITCH 3}) \ 468, 466
468 \text{PRINT 508, } K, \text{ XC}(K)
\text{GO TO 466}
470 \text{IF } (T - \text{CUT}) \ 216, 472, 472
472 \text{IF } (\text{SENSE SWITCH 3}) \ 471, 473
473 \text{PRINT 501, } T
\ K = 1
474 \text{PRINT 504, } K, \text{ TM}(K)
\ K = K + 1
\text{IF } (K - \text{NN}) \ 474, 474, 475
475 \ K = 1
476 \text{PRINT 508, } K, \text{ XC}(K)
\ K = K + 1
\text{IF } (K - \text{IB}) \ 476, 476, 471
471 \text{PRINT 560}
\text{PAUSE}
\text{ACCEPT 505, CUT}
\text{GO TO 470}

\text{C}

CONTINUE

200 CONTINUE
201 T = 0.
   TP = 0.
   KEZ = 2
203 I = 1
204 J = 1
   PDX = 0.
205 PDX = PDX + A(I,J)X(J)
   J = J + 1
   IF (J - NV) 205, 205, 206
206 J = 0
207 IF (J - IB) 206, 215, 215
208 J = J + 1
   PDX = PDX + B(I,J)U(J)
   GO TO 207
215 PDX(I) = PDX
   XX(I) = X(I) + DELTA*PDX
   J = J + 1
   IF (I - NV) 204, 204, 239
216 I = 1
   T = T + DELTA
   TP = TP + DELTA
   J = 0
217 IF (J - NU) 209, 047, 047
209 J = J + 1
   L = INDI(J)
   GO TO (300, 300, 330, 350, 340, 217), L
218 J = 1
   PDX = 0.
219 PDX = PDX + A(I,J)XX(J)
   J = J + 1
   IF (J - NV) 219, 219, 220
220 J = 0
222 IF (J - IB) 223, 226, 226
223 J = J + 1
214 PDX = PDX + B(I,J)U(J)
   GO TO 222
226 X(I) = X(I) + 0.5*(DX(I) + PDX)*DELTA
   J = J + 1
   IF (I - NV) 218, 218, 203
701 CONTINUE

501 FORMAT (/4H T =, E11.5, 5H, SEC)
502 FORMAT (/3H 1,3X,13HVOLTAGE '1' - 0,3X, 1941-TH BRANCH CURRENT)
503 FORMAT (8X, 5HVOLTS, 17X, 4HAMPS)
504 FORMAT (1HV, 13, 2X, E15.8)
505 FORMAT (E15.8)
508 FORMAT (1HC, 13, 15X, E15.8)
540 FORMAT (25HTRANSIENT ANALYSIS PASS 3)
560 FORMAT (45HTO EXTEND TIME,PUSH START,TYPE NEW FINAL
         1 TIME)

C
   END

C
APPENDIX IV

In the future it may be desired to add certain features, such as sensitivity calculations or worse-case analysis, which depend upon the voltages and currents computed in existing programs. These new features may be programmed as a second pass which would be called automatically by the existing pass.

Any data to be transmitted from pass one to pass two must be in a common area, specified by a COMMON statement at the start of each source program. In the pass-one source program two PAUSE statements must be placed at the point where the second pass is to be called. Pass one should then be recompiled using the PDQ compiler with sense switch two turned off. The source deck will be punched along with the object deck. Sense switch three also should be left off so that the PDQ-Free Format subroutines can be compiled into the object deck. By looking at a printed listing of these punched cards, the instructions generated by the double PAUSE commands may be located. These instructions, M8xxxxxxxxxxM8xxxxxxxxxxx (where the x's represent arbitrary characters and the M8 is equivalent to the 48 command), should be changed on the card to L60000000500M900000000000 (where the first 12 digits cause a card to be read and the next 12 transfer control to the first instruction on that card). The source deck and object deck may be separated by sorting on the 11's punch
in column 2, a punch which is present only on the source deck
cards. After pass two is compiled, the first two cards of the object
deck must be discarded as these cards zero core.
APPENDIX V

At present the transient analysis does not allow non-zero initial capacitor voltages or inductor currents. In order to specify initial conditions, the following statements should be added to the pass three source deck following statement 045 + 1: PUNCH 503.

READ 500, J
500 FORMAT (I3)
044 IF (J) 043, 047, 047
043 READ 506, I, X(I)
506 FORMAT (I3, E15.8)

J = J - 1

GO TO 044

After recompiling the third pass using the PDQ compiler, the first two cards of the object deck must be discarded.

The initial conditions would then be specified by a series of cards which are read after the third pass has been loaded. The first card contains the number of initial conditions to be specified. The next cards each contain one integer and one floating point field, the first specifying the number of the state variable, the second specifying the initial value of the state variable.

The state variables for a circuit are first the capacitor tree branch voltages listed in the order of block number, and second the
inductor link currents in the order of block number. When a capacitor loop exists in the circuit, the tree does not contain all the capacitors. The initial conditions, however, can be specified only for those capacitors that are actually in the tree. But the capacitor link voltages are functions of the capacitor tree branches so that specifying the latter also specifies the former. A similar argument can be made for the dependency of the inductor tree branch currents on the inductor link currents.