A GUIDE FOR THE CLASSROOM USE OF IBM'S ECAP/360

by

JAMES EDWARD FRASER 265

B. S., Kansas State University, 1962

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1968

Approved by:

Donald H Lenhert
Major Professor

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure — Page

INTRODUCTION

There are two programs available from the IBM Corporation called ECAP (Electronic Circuit Analysis Program); one is for the IBM 1620 computer, the other for IBM System 360 computers. Once element values, circuit connections, and excitations have been specified; ECAP will solve for circuit voltages and currents (and some other parameters of special interest). ECAP will not synthesize a circuit. ECAP/1620 was the first ECAP program, and as such is very well documented. The manuals available from IBM on ECAP/1620 are the ECAP/1620 USER'S MANUAL (1), the ECAP/1620 SYSTEM MANUAL (2), the ECAP/1620 APPLICATION DESCRIPTION (3), and the ECAP/1620 OPERATOR'S MANUAL (4). ECAP/360 is an expanded and improved version of ECAP/1620; but it is a 'Type 3' program. A 'Type 3' program is one that has been contributed to the IBM program library by an IBM employee (in this case, Gerald R. Hogsett); but which is not supported by IBM, and is not completely documented. In general, the ECAP/1620 USER'S MANUAL also applies to ECAP/360; however, some ECAP/360 features are not covered in the USER'S MANUAL. These features are very briefly discussed in the IBM S/360 GENERAL PROGRAM LIBRARY booklet on ECAP/360 (5). This booklet is the only authoritative documentation on ECAP/360 currently available. Some errors in the USER'S MANUAL are discussed in this report.

The IBM manuals on ECAP are very detailed; and not readily available. The last three chapters of this report are complete,

concise instructions for using ECAP/360; and represent a condensation of all currently available IBM ECAP publications. These chapters are intended to be instruction guides for students. The first two chapters of this report are intended to give a class instructor a reasonable insight into the operation of the program. Chapter one covers both ECAP/360 DC and AC analysis method of solution. Only an elementary knowledge of FORTRAN is needed to understand the program extracts included in chapter one. Chapter two covers ECAP/360 Transient analysis method of solution. Both chapters one and two contain a few comments applicable to ECAP/1620.

Chapter three is the first of the instruction chapters and covers ECAP/360 DC analysis. Chapter three is broken into two parts. The first part is basic instructions, and is intended to be given to 'first course' EE students. The second part covers additional solution features that would be of use to advanced students, but would only encourage 'first course' students to try to 'fly without wings'. Chapter three is complete and self-contained; and no additional material from this report or IBM manuals need be given to students. Chapter four, which covers ECAP/360 AC analysis, is arranged exactly like chapter three; again in two parts. It can be given to students who have not previously used ECAP/360.

Chapter five covers ECAP/360 Transient analysis. Although this chapter is self-contained, it would be better to introduce students to either ECAP/360 DC or AC analysis before Transient analysis. Chapter five cannot be broken into parts, as is the

case with chapters three and four. As the situation currently stands, the ECAP/360 Transient analysis program available at the KSU Computing Center will produce inconsistent solutions under some conditions. Until this problem is corrected, chapter five should be restricted to faculty distribution. Chapters three, four, and five contain specific, card-by-card instructions for getting the ECAP/360 program from disc storage at the KSU Computing Center. As these cards may be changed, the instruction guides may have to be 'up-dated' periodically.

ECAP/360 is ideally suited for handling problems in batches. Batches can contain a mixture of DC, AC, and Transient problems. Since Transient problems 'usually' take longer to solve, it should be recommended that they not be batched. An error in the user data could cause a transient problem to use all the time alloted on the 'job-card' without producing meaningful solutions. Batching DC and AC problems could be very useful to an instructor; all student problems could be solved in one batch. Since it takes 'about' one quarter of a minute to get the program from disc storage, considerable computer time could be saved by batching. Some of the 'variation of parameter' techniques available in DC and AC analysis would allow an instructor to assign a 'different' problem to each student, but allow him to solve all the problems with one short data card deck of his own.

ECAP/360 will troubleshoot many of the program user's errors. Some of the error messages pinpoint data card errors

(obvious from the error message), and many other errors are indicated by error number codes. The error number codes are listed in the ECAP/1620 USER'S MANUAL and the S/360 GENERAL PROGRAM LIBRARY booklet.

A complete object deck for ECAP/1620 is available at Kansas State University for the Industrial Engineering Department's IBM 1620 computer. Complete documentation for ECAP/1620 is also available. The instruction guide chapters of this report are not intended to serve as instructions for ECAP/1620. Those persons specifically interested in ECAP/1620 should read chapters one and two and appendix A of this report; and be extremely familiar with all ECAP/1620 manuals.

ECAP is not the only computer program available to assist in electrical circuit design. One issue of ELECTRO-TECHNOLOGY magazine (6) listed and briefly described the capabilities of forty two circuit design programs. Some of these programs will synthesize specific types of networks. Articles on computer aided circuit design appeared in several 1967 issues of ELECTRONICS magazine. The article by Frank Branin (7) in the January 9, 1967, issue of ELECTRONICS listed several methods of solving electrical circuits; and, more importantly, a large number of references.

CHAPTER 1

ECAP'S METHOD OF SOLUTION FOR DC AND AC CIRCUITS

General Considerations

Before an electrical circuit can be converted into a system of algebraic equations, rules and conventions must be adopted that define positive (or, in phase) values. ECAP has predetermined conventions that must be observed. Some of these conventions are:

1. Elements are defined as resistors, inductors, and capacitors.

2. Every element in a circuit must have a finite, non-zero magnitude.

3. Every element must have a 'from' and a 'to' node.

4. Positive (or, in phase) current flows from the 'from' node of an element to it's 'to' node.

5. One node must be assigned the number zero (nϕ), and this node will serve as the zero reference for node voltages.

6. Every element must be assigned a unique, positive non-zero branch number.

7. All node numbers between the lowest and highest must be assigned.

8. No node or group of nodes can be isolated from the reference node.

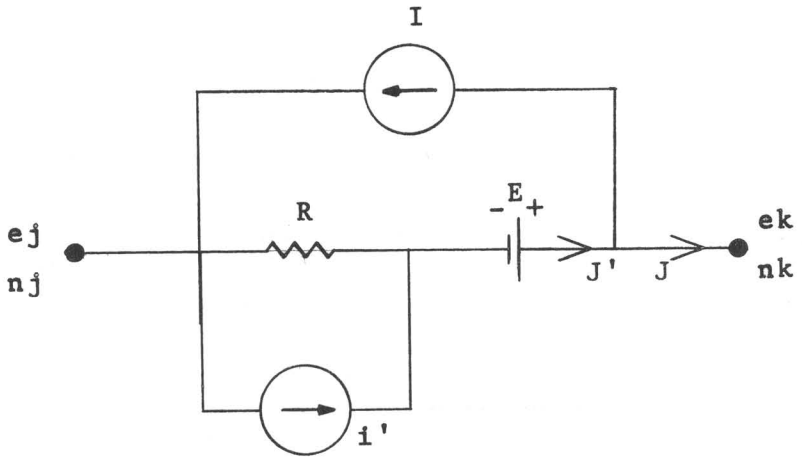
9. A voltage or current source must be associated with a branch (and therefore an element).

All of the preceding conventions can be depicted in a 'standard circuit branch'. Unfortunately, the 'standard circuit branch' for ECAP/360 and ECAP/1620 is not the same as the one shown in the IBM ECAP/1620 USER'S MANUAL (1). Figure 1-1 shows both 'standard branches'. The DC circuit case has been assumed, therefore the element is shown as a resistor. In all cases, I is a fixed-magnitude current source, i' is a current source whose magnitude depends upon the voltage across the element of some other branch, and E is a fixed-magnitude voltage source. Positive conventions are as indicated by arrows and polarity marks. Node numbers are indicated by nj and nk, and ej and ek are node voltages. Further circuit relations can be defined from the standard branches. These relations are summarized in Table 1-1.

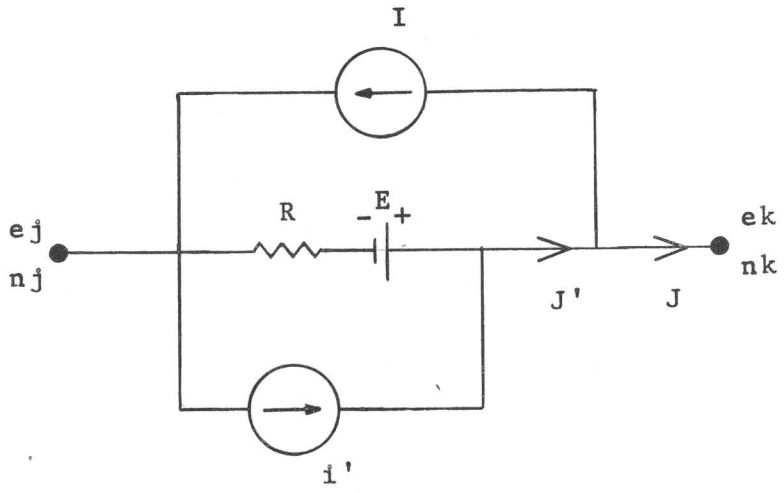
TABLE 1-1

SUMMARY OF ECAP BRANCH RELATIONS

Relation	Symbol	ECAP/360	USER'S MANUAL
Branch Voltage	V	ej - ek	ej - ek
Element Voltage	V'	V+E	V+E
Element Current	J'	(V'/R) + i'	(V'/R)
Branch Current	J	J'-I	J'-I
Element Power Loss	P	J'V'	J'V'



ECAP 360 Branch



User's Manual Branch

Figure 1-1. Comparison of ECAP Standard Branches

The definitions for voltages, currents, and power are the same for both the USER'S MANUAL 'standard circuit branch' and the ECAP/360 'standard circuit branch'. Note, in particular, that element current includes both the resistor current and the dependent source current. From the USER'S MANUAL 'standard circuit branch', a casual observer would conclude that the branch element power loss (excluding both the fixed-magnitude current source and the fixed-magnitude voltage source) would be:

$$P = (V')^2/R + V\ i'$$

since V' acts across the resistor, and V acts across the dependent source. Reference to Table 1-1 shows that power loss for the USER'S MANUAL branch is defined as:

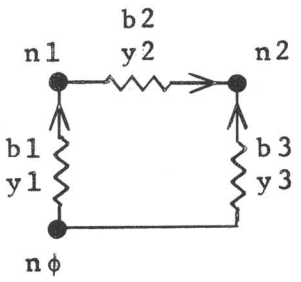
$$P = V'J'$$

or just element current multiplied by element voltage. Thus, the USER'S MANUAL 'standard circuit branch' is not consistant with its power loss definition. For both ECAP/360 and ECAP/1620, power loss is calculated as V'J'; therefore the ECAP/360 'standard circuit branch' used throughout the remainder of this report is valid for both ECAP/360 and ECAP/1620. Since the dependent source current is included in the element current, the power loss calculation includes power transferred into and out of the branch. This can be useful in modeling of active circuit elements.

Construction of Nodal Equations

All equations will be considered to be in matrix or array form. Terms like I(k) are one dimensional arrays with k as a subscript (or index); terms like GM(c,k) are two dimensional arrays with the two subscripts, c and k. Node numbers are indicated by nj where j is the node number. Branch numbers are indicated by bk and element admittances by yk; where k is the branch number. A dependent current source in branch k is indicated by i'(k), and it has an associated transconductance GM(c,k) where the subscript c indicates the control branch. An element voltage of positive magnitude V in branch c causes a positive current of V GM(c,k) to flow in dependent source i'(k). If a current gain BETA(c,k) is specified instead of the transconductance; it is converted by the relation GM(c,k) = BETA(c,k)/R(c), where R(c) is the resistance of branch c. I(k) and E(k) are the fixed-magnitude current and voltage sources associated with branch k.

Figure 1-2 has three examples of simple circuits along with the necessary ECAP/360 convention assignments and nodal equations. Figure 1-2a shows the construction of the nodal admittance matrix equations for a circuit with no exciting sources or dependent sources. Although branch current direction arrows have been assigned, at this point such assignment is arbitrary (a program user's choice) and in no way changes the admittance matrix. A common representation for the nodal admittance matrix would be [Y]; and the matrix equations, [Y][E] = [I]. The nodal admittance

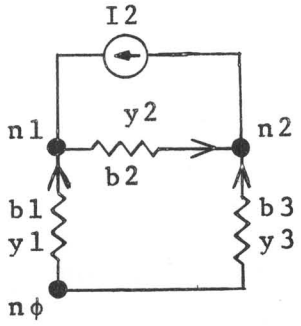


Norton equivalent of branch 2

$$\begin{bmatrix} (y1+y2) & -y2 \\ -y2 & (y2+y3) \end{bmatrix} \begin{bmatrix} e1 \\ e2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

y1 is branch 1 admittance

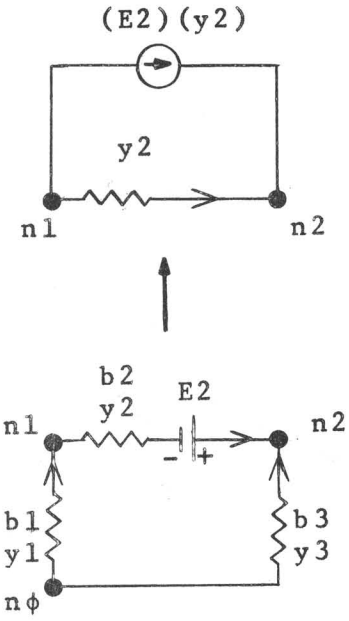
a. No Excitation Sources



$$\begin{bmatrix} (y1+y2) & -y2 \\ -y2 & (y2+y3) \end{bmatrix} \begin{bmatrix} e1 \\ e2 \end{bmatrix} = \begin{bmatrix} I2 \\ -I2 \end{bmatrix}$$

I2 is branch 2 current source

b. Current Source Excitation



Norton equivalent of branch 2

$$\begin{bmatrix} (y1+y2) & -y2 \\ -y2 & (y2+y3) \end{bmatrix} \begin{bmatrix} e1 \\ e2 \end{bmatrix} = \begin{bmatrix} -(E2)(y2) \\ (E2)(y2) \end{bmatrix}$$

E2 is branch 2 voltage source

c. Voltage Source Excitation

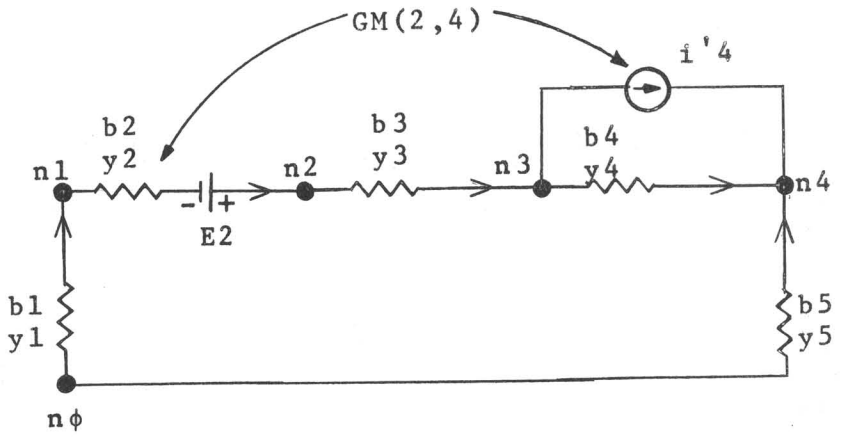
Figure 1-2. Constructing Nodal Equations

matrix will be represented here as [ZPRL] instead of [Y]; and the matrix equations as [ZPRL][E] = [EQUCUR]. [ZPRL] is a double subscripted array where the subscripts are row and column indices arranged in increasing order and in one-to-one correspondence with node numbers. Row k of [ZPRL] is the result of applying Kirchoff's nodal law to the circuit at node k. The reference node nϕ is not included in the system of nodal equations. If NNODE is the largest node number in the circuit; [ZPRL] is an NNODE by NNODE matrix. If branch bk has an admittance of yk, and its nodal connections are 'from' s and 'to' t; then yk is summed into the principal diagonal terms ZPRL(s,s) and ZPRL(t,t), and the negative of yk is summed into the terms ZPRL(s,t) and ZPRL(t,s). The term ZPRL(p,q) is the entry in row p and column q of [ZPRL].

The matrix [E] is a one by NNODE array of the unknown node voltages arranged in increasing order. E(p) is the entry in column p of [E], and is the voltage at node p (also sometimes to be represented in this report as ep). Likewise, the matrix [EQUCUR] has entrys EQUCUR(p). All entrys in [EQUCUR] are known values derived from the program user's data. Figure 1-2b shows that the positive convention current source in branch k is summed into the term EQUCUR(s) and the negative of the source is summed into the term EQUCUR(t); where again s is the 'from' node and t is the 'to' node of branch k. Figure 1-2c indicates that the voltage source in a branch is converted into a Norton equivalent current source, and the equivalent current source is handled

the same as the branch current source in Fig. 1-2b. Notice that the + and - signs in [EQUCUR] of Fig. 1-2c are reversed from Fig. 1-2b because of 'Standard Circuit Branch' positive conventions. Since the entrys in [EQUCUR] are the sum of equivalent current sources attached to each node; EQUCUR as a short name for Equivalent Current Vector (or matrix) is not unreasonable.

Figure 1-3 shows that [ZPRL] is not necessarily symmetric if dependent current sources are included in the circuit. Given GM(j,k); j is the control branch, and k is the source branch. The transconductance GM(j,k) is summed into the term of [ZPRL] with row subscript corresponding to the 'from' node of the dependent source and column subscript corresponding to the 'from' node of the control branch. It is also summed into the term with row subscript corresponding to the 'to' node of the source and column subscript corresponding to the 'to' node of the control branch. Since the 'from-from' and 'to-to' entrys in [ZPRL] get GM(j,k) summed into them, it would be expected that the 'from-to' and 'to-from' entrys would get -GM(j,k) summed into them. Inspection of [ZPRL] in Fig. 1-3 shows that such is the case. The nodes of the source branch determine the rows of the entry, and the nodes of the control branch determine the column of the entry. Since the dependent current source is controlled by the element voltage of the control branch; if the control branch has a fixed-magnitude voltage source, the matrix [EQUCUR] must be adjusted. A voltage 'bias' in the control branch would cause



$$\begin{bmatrix} (y1+y2) & -y2 & 0 & 0 \\ -y2 & (y2+y3) & -y3 & 0 \\ GM(2,4) & -y3-GM(2,4) & (ye+y4) & -y4 \\ -GM(2,4) & GM(2,4) & -y4 & (y4+y5) \end{bmatrix} \begin{bmatrix} e1 \\ e2 \\ e3 \\ e4 \end{bmatrix} =$$

$$= \begin{bmatrix} -(E2)(y2) \\ +(E2)(y2) \\ -(E2)GM(2,4) \\ (E2)GM(2,4) \end{bmatrix}$$

Figure 1-3. Constructing Nodal Equations with Dependent Current Sources

a current 'bias' in the branch of the dependent source. Therefore the row term of [EQUCUR] corresponding to the 'from' node of the source must be adjusted by an amount (Ej)GM(j,k), where Ej is the voltage bias in the control branch. The row term of [EQUCUR] corresponding to the 'to' node of the source must be adjusted by an amount +(Ej)GM(j,k). Figure 1-3 shows an example of adjusting [EQUCUR] for dependent source control branch voltage bias.

Instead of forming [EQUCUR] by considering current sources, voltage sources, and dependent sources separately; it would be much easier to convert the branch sources into one 'big' Norton equivalent source. Then [EQUCUR] could be formed by adding and subtracting branch equivalent sources, and each branch would only have to be examined once for its node connections.

ECAP/360 forms [ZPRL] and [EQUCUR] as arrays by the same procedure outlined in the preceding paragraphs. An extract from ECA 22, the subroutine that forms [ZPRL] for DC circuit analysis is reproduced below.

```
      SUBROUTINE ECA22(ZPRL)
      DO 10 J=1,NNODE
      DO 10 K=1,NNODE
10    ZPRL(J,K)=0.0
      DO 20 I=1,NMAX
      NI=NINIT(I)
      NF=NFIN(I)
      IF (NI) 21,22,21
22    IF (NF) 23,20,23
21    IF (NF) 24,25,24
25    NF = NI                                              I
      GO TO 23
24    ZPRL(NI,NI) = ZPRL(NI,NI) + YX(I)
      ZPRL(NI,NF) = ZPRL(NI,NF) - YX(I)          II
      ZPRL(NF,NI) = ZPRL(NF,NI) - YX(I)
23    ZPRL(NF,NF) = ZPRL(NF,NF) + YX(I)
20    CONTINUE
```

```
      IF(NTERMS)7000,7000,5500
5500  DO 6500 N=1,NTERMS
      LR = IROWT(N)
      LC = ICOLT(N)
      TERM = YTERMX(N)
      I = NITIT(LR)
      IF(I) 6000,6000,5600
5600  J = NFIN(LC)
      IF(J) 5800,5800,5700
5700  ZPRL(I,J) = ZPRL(I,J) - TERM
5800  J = NINIT(LC)
      IF(J) 6000,6000,5900
5900  ZPRL(I,J) = ZPRL(I,J) + TERM
6000  I = NFIN(LR)
      IF(I) 6500,6500,6100
6100  J = NITIT(LC)
      IF(J) 6300,6300,6200
6200  ZPRL(I,J) = ZPRL(I,J) - TERM
6300  J = NFIN(LC)
      IF(J) 6500,6500,6400
6400  ZPRL(I,J) = ZPRL(I,J) + TERM
6500  CONTINUE
7000  RETURN
```

III

where:

NMAX — number of branches in the circuit

NNODE — number of nodes in the circuit,

NINIT — an array of 'from' (initial) node connections of the branches in branch order,

NFIN — an array of 'to' (final) node connections of the branches in branch order,

YX — an ordered array of branch admittances.

In addition, all dependent sources in the circuit are numbered (in order) by the program user; and NTERMS is the total number of dependent sources. All dependent source connections and transconductance data is stored in arrays indexed by dependent source number. These arrays are:

YTERMX — an ordered array of dependent source transconductances,

IROWT an ordered array of dependent source controlled (source) branch numbers,

ICOLT an ordered array of dependent source control branch numbers

The arrays IROWT and ICOLT are aptly named as they are used to determine the branch numbers of the branches that determine the dependent source row and column entrys, respectively, in [ZPRL]. Refer again to [ZPRL] in Fig. 1-3.

In the extract from ECA 22; the statements indicated by bracket I form the symmetric [ZPRL] array, uncorrected for dependent sources. The statements indicated by bracket II sums the branch admittances into the proper terms. Notice, as a quick confirmation, that the terms on the principal diagonal get the admittance added. The statements indicated by bracket III correct [ZPRL] for dependent sources. The signs associated with TERM in the correction can be quickly confirmed by referring to an earlier discussion in this chapter about dependent source conventions and also statement 5900. The RETURN statement at the bottom of the ECA 22 extract directs the program to return to the main line dc analysis control subroutine, ECA 20. Note that if there are no dependent sources, the IF statement between brackets I and III causes execution to skip immediately to statement 7000 (i.e. RETURN).

[EQUCUR] is as easily formed by subroutine ECA 23 as [ZPRL] was by ECA 22. The only variables not previously defined are EX and AMPX. EX is an in order array of branch voltage source magnitudes, and AMPX is an array of branch current source

(exclusive of dependent current sources) magnitudes. All other variables are self-explanatory by their usage. An extract from ECA 23 is included below.

```
      SUBROUTINE ECA 23
      DO 4000 LL = 1,NMAX
4000  CURR(LL) = YX(LL) * EX(LL) - AMPX(LL)
      IF (NTERMS) 9000,9000,7500
7500  DO 8000 I = 1,NTERMS                                   IV
      L = ICOLT(I)
      LL = IROWT(I)
8000  CURR(LL) = CURR(LL) + YTERMX(I) * EX(L)
9000  DO 9100 K = 1,NNODE
9100  EQUCUR(K) = 0.0
      DO 9500 LL = 1,NMAX
      II = NINIT(LL)                                          V
      JJ = NFIN(LL)
      IF (II) 9300,9300,9200
9200  EQUCUR(II) = EQUCUR(II) - CURR(LL)
9300  IF (JJ) 9500,9500,9400
9400  EQUCUR(JJ) = EQUCUR(JJ) + CURR(LL)
9500  CONTINUE
      RETURN
```

The statements indicated by bracket IV in the extract from ECA 23 converts all branch sources into one Norton equivalent current source of negative convention. This was indicated as a possibility in an earlier discussion in this chapter. The statements indicated by bracket V forms the terms of [EQUCUR] from the equivalent branch sources.

Solution of DC Circuit Nodal Equations

ECAP/360 solves its DC nodal matrix equation [ZPRL][E] = [EQUCUR] by inverting [ZPRL] and then premultiplying both sides of the equation by this inverse. This process yields [E] = $[ZPRL]^{-1}$[EQUCUR], and [E] is the desired solution for node

voltages. $[ZPRL]^{-1}$, the nodal impedance matrix, is available as part of the solution outputs if desired. Subroutine ECA 26 performs the required inversion and matrix multiplication. Only two new definitions are necessary:

N is the number of independent nodes (earlier called NNODE),

DABS is the double precision absolute value function,

SMLEP is an ordered array of solution node voltages.

All calculations in forming the inverse of [ZPRL] are performed in double precision. An extract from ECA 26 is reproduced below.

```
      SUBROUTINE ECA26(ZPRL)
      N = NNODE
3     DO 20 J=1,N
20    IPIV(J) = 0.0
      DO 550 I = 1,N
      AMAX = 0.0
      DO 105 J = 1,N
      IF (IPIV(J) - 1) 60,105,60
60    DO 100 K = 1,N
      IF (IPIV(K) - 1) 80,100,750                                   VI
80    IF (DABS(ZPRL(j,K)) - DABS(AMAX)) 100,100,85
85    IROW = J
      ICOL = K
      AMAX = ZPRL(J,K)
100   CONTINUE
105   CONTINUE
      IPIV(ICOL) = IPIV(ICOL) + 1
      IF (IROW - ICOL) 140,260,140
140   DO 200 L = 1,N
      SWAP = ZPRL(IROW,L)
      ZPRL(IROW,L) = ZPRL(ICOL,L)                                   VII
200   ZPRL(ICOL,L) = SWAP
260   INDEX(I,1) = IROW
      INDEX(I,2) = ICOL
      ZPRL(ICOL,ICOL) = 1.0
      DO 350 L = 1,N                                                VIII
```

```
350   ZPRL(ICOL,L) = ZPRL(ICOL,L)/AMAX
      DO 550 LI = 1,N                                          VIII
      IF (LI - ICOL) 400,550,400
400   T = ZPRL(LI,ICOL)
      ZPRL(LI,ICOL) = 0.0
      DO 450 L = 1,N
450   ZPRL(LI,L) = ZPRL(LI,L) - ZPRL(ICOL,L) * T
550   CONTINUE
      DO 710 I = 1,N
      L = N + 1 - I
      IF (INDEX(L,I) - INDEX(L,2)) 630,710,630
630   IROW = INDEX(L,1)
      ICOL = INDEX(L,2)                                        IX
      DO 705 K = 1,N
      SWAP = ZPRL(K,IROW)
      ZPRL(K,IROW) = ZPRL(K,ICOL)
      ZPRL(K,ICOL) = SWAP
705   CONTINUE
710   CONTINUE
750   CONTINUE
      DO 4 I = 1,N                                             X
      SMLEP(I) = 0.0
      DO 4 J = 1,N
4     SMLEP(I) = SMLEP(I) + ZPRL(I,J) * EQUCUR(J)
9999  RETURN
```

In ECA 26; the first time the DO 550 loop following statement 20 is executed, the statements included in bracket VI locate the largest term in [ZPRL]. The statements in bracket VII put the largest term on the principal diagonal by interchanging rows. The INDEX terms in statement 260 and the following statement allows the subroutine to remember what rows were interchanged. The statements in bracket VIII start an inversion process that is similar to synthetic (Gaussian) elimination. Synthetic elimination would insert an identity matrix of rank N 'to the right' of [ZPRL], and then develop zeros in off-diagonal terms of [ZPRL], and ones on the principal diagonal. The zeros would be developed by elementary row transformations. An elementary

row transformation is an operation on a matrix that interchanges any two rows, or multiplys any row by a non-zero constant, or adds any row to another row. The zeros in off-diagonal terms would be developed by Gaussian elimination first below the principal diagonal (one column at a time), and then above the diagonal (still above the diagonal and within the rows and columns of the original [ZPRL]). ECA 26 differs from this process in a few details. It doesn't specify the location of the unity entrys in the identity matrix until the AMAX term has been placed on the principal diagonal of [ZPRL]. It doesn't physically carry the identity matrix as part of the array. It also develops zeros in all terms of one column of [ZPRL] (exclusive of the diagonal term which is unity) at a time instead of first above and then below the diagonal. This process of inversion is executed in the statements of bracket VIII. Note that the column of [ZPRL] that would become zeros in the process of transformation are pulled out and stored as 'T', and the similarly located terms of the identity matrix (all zeros already) are substituted in their place before the transformation. This gives the same result as carrying the identity matrix terms in the array. Note, too, that the unity entry of the identity matrix was substituted in [ZPRL] two statements before statement 350. This gives a term 1/AMAX on the principal diagonal. The inversion continues by returning to bracket VI and locating the next largest element in the array after excluding the column that was just substituted and the row that was the basis of the transformations. When DO 550 I = 1,N

is completed, an inverse has been constructed; but not the inverse of [ZPRL]. The inverse of [ZPRL] is constructed from previous results by re-interchanging rows in the reverse order in which they were interchanged by the statements of bracket VI. The rows are 'deswapped' in the statements of bracket IX.

The solution for node voltages is obtained by premultiplying [EQUCUR] by the inverse of [ZPRL]. This is the result of the statements of bracket X.

Subroutine ECA 25 computes and controls the print out of branch voltages, element voltages, element currents, branch power losses, and branch currents. An extract of ECA 25 that omits the print out instructions is included below.

```
      SUBROUTINE ECA 25
C     BRANCH VOLTAGES
101   DO 10 I = 1,NMAX
      SMLE(I) = 0.0
      J = NINIT(I)
      IF (J) 11,11,12
12    SMLE(I) = SMLEP(J)
11    K = NFIN(I)
      IF (K) 10,10,14
14    SMLE(I) = SMLE(I) - SMLEP(K)
10    CONTINUE
c     branch voltages smle(i) are printed out
C     ELEMENT VOLTAGES
102   DO 17 I = 1,NMAX
17    SMLE(I) = SMLE(I) + EX(I)
c     element voltages smle(i) are printed out
C     ELEMENT CURRENTS
103   DO 20 I = 1,NMAX
20    CURR(I) = YX(I) * SMLE(I)
      IF (NTERMS) 21,22,21
21    DO 23 I = 1,NTERMS
      NR = IROWT(I)
      NC = ICOLT(I)
      IF (SMLE(NC)) 24,23,24
```

```
24    CURR(NR) = CURR(NR) + YTERMX(I) * SMLE(NC)
23    CONTINUE
c     element currents curr(nr) are printed out
C     BRANCH POWER LOSSES
28    DO 29 I = 1,NMAX
29    X(I) = CURR(I) * SMLE(I)
c     branch power losses x(i) are printed out
C     BRANCH CURRENTS
105   DO 30 I = 1,NMAX
30    CURR(I) = CURR(I) - AMPX(I)
c     branch currents curr(i) are printed out
```

Liberty has been taken to insert a small case comment in the extract from ECA 25 to show where the output controls have been omitted. These comments do not appear in the original. In the calculation for element current, the values of the variable CURR are computed and are not the same values used with a variable of the same name in computing [EQUCUR] in subroutine ECA 23. Note that the calculations for element current includes both the dependent source current and the current in the element. Then in the calculations for branch power losses, the above calculated values for element current is multiplied by element voltage. This calculation for power loss justifies the 'Standard Circuit Branch' specified for ECAP 360 earlier in this chapter.

Branch currents are calculated in ECA 25 as the sum of all currents in the branch. In a set of calculation that has been omitted from the extract, the branch currents are summed at each node. The absolute magnitude of the difference from zero at each node is summed over all nodes. This absolute magnitude sum represents the current unbalance in the solution and is a measure of the accuracy of the solution. If this unbalance exceeds a user

specified amount (or in the absence of user instructions, a program default amount of .001 ampere), the statement "SOLUTION NOT OBTAINED TO DESIRED TOLERANCE" will be printed along with the obtained solution. ECAP/1620 does not have this current unbalance check in its DC analysis program.

Calculation of Partial Derivatives of Node Voltages for DC Circuits

Circuit design is not complete once the connections and nominal element values are specified. No two elements with the same nominal value have exactly the same value, but the expected amount of deviation from the nominal value can be specified. In many circuits, certain key element values influence the voltage at a given node more than other element values. It is reasonable to specify these key elements to be of a closer tolerance.

The key elements for a specific node voltage can be located by evaluation of the partial derivatives of the node voltage with respect to each element at the nominal value of these elements. The node voltage is more sensitive to change in value of the element with the largest partial derivative. The partial derivative gives the designer an estimate of how much node voltage will vary with a one unit change of element value. Element values are commonly specified by a nominal value and a percentage deviation, therefore it would be more convenient if the partial derivatives could be adjusted to reflect a given percentage of deviation from the nominal value. This is accomplished by multiplying the

partial derivative by the nominal value divided by one hundred. The result is called a "Sensitivity Coefficient", and is the voltage variation due to a one percent deviation in element value. The key elements are those with the largest sensitivity coefficients.

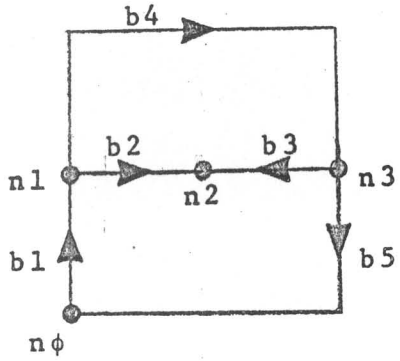
Once the nominal value and tolerance of each element has been specified, it is necessary to estimate or calculate the largest and smallest voltage that could occur at each node (or maybe only at one node of interest). ECAP defines these cases to be the 'WORST CASE MAX' and 'WORST CASE MIN' respectively. For example, the worst case max. for node seven is a solution of the nodal equations for node seven's voltage. For this solution; the maximum allowable value of all elements with positive partial derivatives at node seven is substituted in the equations in place of the nominal values, and minimum values for all elements with negative partial derivatives. The solution for all worst case node voltages requires the solution of the nodal equations twice for each independent node. To save time, ECAP/360 allows the user to specify the nodes for which the worst case solution is desired. ECAP/1620 does not allow the user to specify worst case nodes, instead it solves for all worst cases.

It should be remembered that the partial derivatives are not constants, and they can change sign with changes in nominal element values. After solving for a worst case, ECAP/360 recalculates the partials. If any partial has changed sign, an error message will be printed. However, if no error message is

printed; the worst case solution is exact (not an approximation). ECAP/1620 assumes that all partials are constant and independent. Its worst case solution is a sum of approximate deviations calculated from the partials; and the solution is approximate. ECAP/1620 does not check the signs of the partials at each worst case.

Before discussing ECAP calculation of circuit sensitivities and partial derivatives, a bit of network topology is needed. If all node voltages are known for a given circuit, the voltage across a branch is defined (positive convention) as the 'from' node voltage minus the 'to' node voltage. If the node voltages are ordered (excluding the reference node) in a column matrix, it is possible to represent the conversion to a column matrix of branch voltages as a premultiplication of the node voltage matrix by a properly devised incidence matrix. This incidence matrix has a row for each branch in the circuit and the rows are in branch order. Thus the result of multipling row j of the incident matrix and the node voltage matrix, the voltage across branch j. Row j of the incidence matrix can have (at most) only two non-zero entrys; a (+1) in the column corresponding to the 'from' node of branch j, and a (-1) in the column corresponding to the 'to' node. Less than two non-zero entrys can occur only if either the 'from' or 'to' node of a branch is the reference node (nϕ). The incidence matrix must have the same number of columns as the circuit has independent nodes in order that the two matrices are conformable for multiplication. If [A] is the incidence matrix, [V] is the node

voltage column matrix, and [V'] is the branch voltage column matrix; then [V'] = [A][V]. An example of a circuit and corresponding incidence matrix is given in Fig.1-4. Some reference sources (8) define the incidence matrix to be the transpose of the incidence matrix defined here, but this is no more than a change of notation.



$$V' = \begin{array}{c} \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{array} \begin{array}{c} \begin{array}{ccc} n1 & n2 & n3 \end{array} \\ \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} \\ V1 \\ \\ V2 \\ \\ V3 \\ \\ \end{bmatrix} = [A][V]$$

$$V'2 = V1 - V2$$

Fig. 1-4. Example Circuit and Incidence Matrix

Define a square matrix [Y] with the number of rows (and columns) equal to the number of branches of the circuit, diagonal terms like Y(j,j) = yj (the admittance of branch j), and of diagonal terms Y(i,j) equal to the transconductance (if any) between branch j and branch i (voltage across j causes current in i dependent source). If the column matrix of branch current sources is [I], and the column matrix of branch voltage sources is [E]; then it can be shown that:

$$[A^tYA][V] = [A^t][I - YE] \qquad (1)$$

In Eq. 1 and in the following derivations, the superscript t

indicates the transpose of the preceding matrix, and the brackets that indicate matrix quantities have been eliminated except where absolutely necessary to indicate order of operations. The superscript -1 indicates the matrix inverse of the preceding quantity. Equation 1 is a complete expression for the circuit nodal equations; and the term $[A^tYA]$ can be identified as the nodal admittance matrix [ZPRL]. It follows that:

$$[V] = [A^tYA]^{-1}[A^t][I - YE] \qquad (2)$$

The term $[A^tYA]^{-1}$ is the nodal impedance matrix for the circuit. In passing, it might be noted that ECAP/1620 forms the system of circuit nodal equations by the matrix multiplication process indicated in Eq. 1.

Since the solution for node voltages in Eq. 2 is a matrix equation; matrix differentiation and a few matrix differentation identities must be introduced (9). Given a matrix [U] with elements U(j,k) and a variable t; then:

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial t}\begin{bmatrix} U(1,1) & U(1,2) & \cdots\cdots \\ U(2,1) & U(2,2) & \cdots\cdots \\ U(3,1) & \cdots\cdots\cdots\cdots & \\ \cdots\cdots\cdots\cdots\cdots\cdots & & \\ U(m,1) & \cdots\cdots\cdots\cdots & U(m,n) \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial U(1,1)}{\partial t} & \cdots\cdots\cdots & \frac{\partial U(1,n)}{\partial t} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \\ \frac{\partial U(m,1)}{t} & \cdots\cdots\cdots & \frac{\partial U(m,n)}{\partial t} \end{bmatrix}$$

If [U], [V], and [W] are matrices conformable for addition, then:

$$\frac{\partial[U + V + W]}{\partial t} = \frac{\partial[U]}{\partial t} + \frac{\partial[V]}{\partial t} + \frac{\partial[W]}{\partial t}$$

If [R], [S], and [T] are matrices conformable for multiplication in that order, then:

$$\frac{\partial[RST]}{\partial t} = \frac{\partial[R]}{\partial t}[ST] + [R]\frac{\partial[S]}{\partial t}[T] + [RS]\frac{\partial[T]}{\partial t}$$

From the differentiation of a matrix product, it is seen that order of the multiplication within individual terms of the sum must be maintained. Also, note that the derivative of a matrix with respect to a variable not included in any of the elements of the matrix is a null matrix, $[\phi]$, of the same order as the original matrix.

ECAP solutions for partial derivatives, sensitivity coefficients, and worst case are available only for DC circuits. The partial derivative of the node voltages with respect to the resistance of branch j, Rj, can be found from Eq. 2; but first note that:

$$[A^tYA]^{-1}[A^tYA] = [U] \quad \text{(U is an identity matrix)}$$

and that

$$\frac{\partial[A^tYA]^{-1}[A^tYA]}{\partial\, Rj} = \frac{\partial[A^tYA]^{-1}}{\partial\, Rj}[A^tYA] + [A^tYA]^{-1}\frac{\partial[A^tYA]}{\partial Rj}$$

$$= [\phi] \quad \text{(the null matrix)} \qquad (3)$$

From Eq. 2, taking the indicated partial:

$$\frac{\partial[V]}{\partial\ Rj} = \frac{\partial[A^tYA]^{-1}}{\partial\ Rj}\ [A^t][I - YE] + [A^tYA]^{-1}\ \frac{\partial[A^t][I - YE]}{\partial\ Rj}$$

Since neither I nor E depend upon Rj:

$$\frac{\partial[V]}{\partial\ Rj} = \frac{[A^tYA]^{-1}}{\partial\ Rj}\ [A^t][I - YE] - [A^tYA]^{-1}[A^t]\ \frac{\partial[Y]}{\partial\ Rj}\ [E]$$

Substituting from the left hand side of Eq. 1:

$$\frac{\partial[V]}{\partial\ Rj} = \frac{\partial[A^tYA]^{-1}}{\partial\ Rj}\ [A^tYA][V] - [A^tYA]^{-1}[A^t]\ \frac{\partial[Y]}{\partial\ Rj}\ [E]$$

Substituting from Eq. 3:

$$\frac{\partial[V]}{\partial\ Rj} = -[A^tYA]^{-1}[A^t]\ \frac{\partial[Y]}{\partial\ Rj}\ [A][V]$$

$$-[A^tYA]^{-1}[A^t]\ \frac{\partial[Y]}{\partial\ Rj}\ [E]$$

Since (mentioned earlier) $[V'] = [A][V]$, then

$$\frac{\partial[V]}{\partial\ Rj} = -[A^tYA]^{-1}[A^t]\ \frac{\partial[Y]}{\partial\ Rj}\ [V' + E] \qquad (4)$$

Equation 4 is much easier to use than the original form, especially when it is realized that Rj appears in only one term in [Y] (that term is on the diagonal).

The partial derivative of the node voltages with respect to the transconductance between branches j and k is derived exactly the same way as the partial with respect to branch resistance.

The result of this derivation is shown in Eq. 5:

$$\frac{\partial [V]}{\partial GM_{jk}} = - [A^t YA]^{-1}[A^t] \frac{\partial [Y]}{\partial GM_{jk}} [V' + E] \qquad (5)$$

where GM_{jk} is the transconductance between branch j and branch k. Since transconductance and current gain are related by the formula $GM_{jk} = BETA_{jk}/Rk$; the partial with respect to $BETA_{jk}$ is:

$$\frac{\partial [V]}{\partial BETA_{jk}} = - [A^t YA]^{-1}[A^t] \frac{\partial [Y]}{\partial BETA_{jk}} [V' + E]$$

ECAP calculates the partial with respect to GM, but not with respect to BETA.

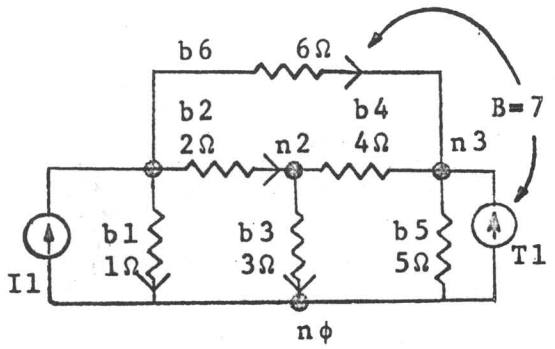
If it is remembered that [I] is the only term in Eq. 2 that depends on Ij (the fixed-magnitude current source in branch j), then the partial of node voltage with respect to Ij is by inspection:

$$\frac{\partial [V]}{\partial Ij} = [A^t YA]^{-1}[A^t] \frac{\partial [I]}{Ij} \qquad (6)$$

Likewise, the partial of node voltage with respect to Ej (the fixed-magnitude voltage source in branch j) is by inspection:

$$\frac{\partial [V]}{\partial Ej} = - [A^t YA]^{-1}[A^t Y] \frac{\partial [E]}{\partial Ej} \qquad (7)$$

For actual calculations of the partials in Eq. 6 and Eq. 7; the indicated inverse and branch voltages are already available to ECAP from earlier execution of subroutines ECA 25 and ECA 26.



I1 = 1 ampere
R1 = 1 mho
R2 = 2 mho
R3 = 3 mho
R4 = 4 mho
R5 = 5 mho
R6 = 6 mho
GM(6,5) = 7(6) mho

$$[ZPRL][SMLEP] = [EQUCUR]$$

$$[A] = \begin{array}{c} \\ B1 \\ B2 \\ B3 \\ B4 \\ B5 \\ B6 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \end{array}$$

$$[Y] = \begin{array}{c} \\ B1 \\ B2 \\ B3 \\ B4 \\ B5 \\ B6 \end{array} \begin{array}{c} \begin{array}{cccccc} B1 & B2 & B3 & B4 & B5 & B6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 42 & 6 \end{bmatrix} \end{array}$$

$$[ZPRL] = [A]^T[Y][A] = \begin{array}{c} \\ n1 \\ n2 \\ n3 \end{array} \begin{array}{c} \begin{array}{ccc} n1 & n2 & n3 \end{array} \\ \begin{bmatrix} 9 & -2 & -6 \\ -2 & 9 & -4 \\ -48 & -4 & 57 \end{bmatrix} \end{array}$$

$$[ZPRL]^{-1} = \frac{1}{1221}\begin{bmatrix} 497 & 138 & 62 \\ 306 & 225 & 48 \\ 440 & 132 & 77 \end{bmatrix}$$

$$[EQUCUR] = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$[SMLEP] = [V] = [ZPRL]^{-1}[EQUCUR] = \frac{1}{1221}\begin{bmatrix} 497 \\ 306 \\ 440 \end{bmatrix}$$

Fig. 1-5. Example Calculation of a Partial Derivative.

$$[SMLE] = [V'] = [A][V] = \frac{1}{1221}\begin{bmatrix} 497 \\ 191 \\ 306 \\ -134 \\ -440 \\ 57 \end{bmatrix}$$

$$[E] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \quad [V' + E] = [V']$$

$$\frac{\partial [Y]}{\partial R5} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial [V]}{\partial R5} = - [ZPRL]^{-1} [A]^T \frac{\partial [Y]}{\partial R5} [V' + E]$$

$$= \frac{-1}{1221} \begin{bmatrix} 497 & 138 & 62 \\ 306 & 225 & 48 \\ 440 & 132 & 77 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \frac{1}{1221} \begin{bmatrix} 497 \\ 191 \\ 306 \\ -134 \\ -440 \\ 57 \end{bmatrix}$$

Fig. 1-5 (cont.) Example Calculation of a Partial Derivative.

$$= \frac{-1}{(1221)(1221)} \begin{bmatrix} 497 & 138 & 62 \\ 306 & 225 & 48 \\ 440 & 132 & 77 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -25 & 0 \end{bmatrix} \begin{bmatrix} 497 \\ 191 \\ 306 \\ -134 \\ -440 \\ 57 \end{bmatrix}$$

$$= \frac{-1}{(1221)(1221)} \begin{bmatrix} 497 & 138 & 62 \\ 306 & 225 & 48 \\ 440 & 132 & 77 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1100 \end{bmatrix} = \frac{-1100}{(1221)(1221)} \begin{bmatrix} 62 \\ 48 \\ 77 \end{bmatrix}$$

$$\frac{\partial [V]}{\partial R5} = \begin{bmatrix} -.457 \\ -.354 \\ -.568 \end{bmatrix}$$ (Solution for all nodes

Fig. 1-5 (cont.) Example Calculation of a Partial Derivative.

The only term that has not already been computed (or stored) is the individual partials; and these result in only one or two entrys in the proper matrix (remember that, for example, $\partial Ij/\partial Ik = 0$). Figure 1-5 is an example calculation of a partial derivative of node voltage with respect to a branch resistance.

Calculation of Standard Deviation of Node Voltages for DC Circuits

A useful theorem from statistical theory (10) is:

> Let X1,X2, ... ,Xn be n mutually stochastically independent random variables having the normal distributions $N(\mu_1,\sigma_1^2)$, $N(\mu_2,\sigma_2^2)$, ... , $N(\mu_n,\sigma_n^2)$ The random variable $Y = k_1X1 + k_2X2 + \ldots + k_nXn$, where $k_1,k_2,k_3, \ldots ,k_n$ are real constants, is normally distributed with mean $k_1\mu_1+k_2\mu_2+\ldots + k_n\mu_n$ and variance $k_1\sigma_1^2 + k_2\sigma_2^2 + \ldots + k_n\sigma_n^2$.

If it is assumed that the partial derivatives of node voltages with respect to element values are independent and constant, then for small element variations the change in node voltages due to these changes can be estimated by

$$\Delta V_i \simeq \sum_{j=1}^{b} \frac{\partial V_i}{\partial R_j} \Delta R_j + \sum_{j=1}^{b} \frac{\partial V_i}{\partial E_j} \Delta E_j + \sum_{j=1}^{n} \frac{\partial V_i}{\partial I_j} \Delta I_j$$

$$+ \sum_{j=1}^{b} \sum_{k=1}^{b} \frac{\partial V_i}{GM_{gk}} \Delta GM_{jk}$$

and the variables are defined as:

ΔV_i = the incremental change of node i voltage.

ΔR_j = the incremental change of branch j resistance.

ΔE_j = the incremental change of branch j source voltage.

ΔI_j = the incremental change of branch j source current.

ΔGM_{jk} = the incremental change of transconductance from branch j to branch k.

b = the total number of branches.

If it is assumed that element values are independent and normally distributed, the means of the distributions are the nominal element values, and that the tolerance figures represent three times the variances of the distributions; then the cited theorem can be applied to estimate the standard deviations of the node voltages. The partial derivatives become the k_j of the theorem. The standard deviation of an individual element can be calculated from the tolerance values by the equation:

$$6\sigma = W_j \left(2 T_j / 100 \right)$$

where:

σ = the element standard deviation.

W_j = the nominal element value.

T_j = the percent tolerance of the element.

From the equation for variance in the theorem, the standard deviation of node voltage, σ_{Vi}, is seen to be:

$$\sigma_{Vi}^2 = \frac{1}{36} \sum_{j=1}^{b} \left[(6\sigma_{Rj})^2 \left(\frac{\partial Vi}{\partial R_j}\right)^2 + (6\sigma_{Ej})^2 \left(\frac{\partial Vi}{\partial Ej}\right)^2 \right.$$

$$\left. + (6\sigma_{Ij})^2 \left(\frac{\partial Vi}{\partial Ij}\right)^2 \right] + \sum_{j=1}^{b} \sum_{k=1}^{b} (6\sigma_{GM_{jk}})^2 \left(\frac{\partial Vi}{\partial GM_{jk}}\right)^2$$

and

$$\sigma_{Vi} = + \sqrt{\sigma_{Vi}^2}$$

If element variation data is specified by maximum, minimum and nominal values; subtraction of the minimum value from the maximum value results in an estimate of a 6σ deviation in element value. This estimate is reasonable since it is 99.8% probable that element value will be inside a $\pm 3\sigma$ interval about its nominal value.

A normal distribution is symmetric with respect to its mean. ECAP assumes that the maximum and minimum values provided by the user are symmetric with the nominal value. If the data is not symmetric, ECAP will produce an 'answer' and call it standard deviation; but clearly the theorem does not apply since the element value is not normally distributed.

Both ECAP 360 and ECAP 1620 solve for node voltage standard deviations by the same method. Since the partial derivatives are assumed to be constant and independent; the solutions are approximations. The approximation is reasonable for small element tolerances; but the definition of what is reasonable or small varies from circuit to circuit. For critical circuits, circuit parameters could be changed by the user. Observing the partial derivatives with the changed values would serve to check their constant and independent nature.

ECAP AC Analysis

ECAP solves single frequency AC circuits by calculating node voltage from a set of simultaneous nodal equations. After converting inductors and capacitors to impedances, the nodal equations are constructed by the same process as were the nodal equations for DC Analysis; but with the added complexity of complex numbers. ECAP stores complex numbers in two arrays for each variable; one for the real part, and the other for the imaginary part. Where the DC Analysis program called the nodal conductance matrix [ZPRL], the AC Analysis program calls the nodal admittance matrix by its two parts, [DUM1] and [DUM2]. The equivalent current matrix, [EQUCUR], becomes [EQUCRL] and [EQUCIM]; and the node voltage matrix, [SMLEP] becomes [EPRL] and [EPIM]. The nodal matrix equation that was solved by DC Analysis was:

$$[ZPRL][SMLEP] = [EQUCUR]$$

but in AC Analysis it becomes:

$$[[DUM1] + j[DUM2]][[EPRL] + j[EPIM]] = [[EQUCRL] + j[EQUCIM]]$$

The AC Analysis standard circuit branch is the same as shown in Fig. 1-1, but the elements can be inductors, capacitors and resistors. The polarity markings indicate the zero degree, in-phase convention.

ECAP AC Analysis will provide the user with solutions for node voltages, branch voltages, element currents, branch currents, and branch power losses. Definitions for these quantities are the same as they were in DC Analysis. AC Analysis will not provide partial derivatives, sensitivity coefficients, worst case solutions, or standard deviation solutions.

The AC Analysis program does have one unique feature, other than some added complexity. Instead of solving the nodal equations by inverting the nodal matrix, it uses a method presented by P. D. Crout (11) in the AIEE Transactions. Figure 1-6 shows an example of a system of equations with real coefficients solved by this method. An array is constructed with elements corresponding to like position entrys in the augmented matrix of the system (See Fig. 1-6b). The solution method is outlined in two sets of rules. The result of operating on the given matrix with the first set of rules is called the auxiliary matrix. The result of operating on the auxiliary matrix with the second set of rules is called the final matrix. The final matrix is the tabulated solutions for the unknown variables. The first set of rules from Crout's paper is:

1. The various numbers, or elements, are determined in the following order: elements of first column, then elements of first row to right of first column; elements of second column below first row, then elements of second row to right of second column; elements of third column below second row, then elements of third row to right of third column; and so on until all elements are determined.

2. The first column is identical with the first column of the given matrix. Each element of the first row except the first is obtained by dividing the corresponding element of the given matrix by that first element.

3. Each element on or below the principal diagonal is equal to the corresponding element of the given matrix minus the sum of those products of elements in its row and corresponding elements in its column (in the auxiliary matrix) which involve only previously computed elements.

4. Each element to the right of the principal diagonal is given by a calculation which differs from rule 3 only in that there is a dinal division by its diagonal element (in the auxiliary matrix).

Figure 1-6c shows an application of the preceding rules to the system of equations of Fig. 1-6a. Again, the result of the first set of rules is the auxiliary matrix. The second set of rules from Crout's paper is:

1. The elements are determined in the following order: last, next to last, second from last, third from last, etc.

2. The last element is equal to the corresponding elements of the last column of the auxiliary matrix.

3. Each element is equal to the corresponding elements of the last column of the auxiliary matrix minus the sum of those products of elements in its row in the auxiliary matrix and corresponding elements in its column in the final matrix which involve only previously computed elements.

Figure 1-6d shows an application of the second set of rules to the auxiliary matrix found in Fig. 1-6c. Notice that the end result of Fig. 1-6d is the solution of the system of equations. This application of Crout's method does not produce an inverse

matrix as a by-product; but for this sacrifice, the number of calculations is reduced. The number of calculations is less than that required for Gaussian elimination. The introduction of complex numbers complicates the procedure by requiring complex multiplication and division, but the method is not changed.

No extracts from the AC Analysis program are included in this report since they are straight forward extentions of the DC Analysis program and the preceding discussion. ECAP/360 has the feature of summing currents at the nodes and calculating the solution current unbalance. In AC analysis, the real and imaginary current unbalances are computed and compared with the user specified maximum unbalance separately. If either the real or imaginary current unbalance exceeds the maximum, an error message will be printed. In the absence of user instructions for the maximum current unbalance, ECAP/360 assumes a default amount of .001 ampere. The real and imaginary maximum current unbalance cannot be specified separately. ECAP/1620 does not have this current unbalance check in its AC Analysis program.

$$3X + 7Y + 6Z = 9$$

$$X - 8Y + 5Z = 0$$

$$5X + 0Y + 4Z = 2$$

a. Original System of Equations

$$\left[\begin{array}{ccc:c} 3 & 7 & 6 & 9 \\ 1 & -8 & 5 & 0 \\ 5 & 0 & 4 & 2 \end{array}\right]$$

Vertical Dots indicate Partitioning

b. Augmented (Given) Matrix of the System

$$\left[\begin{array}{ccc:c} 3 & 7 & 6 & 9 \\ 1 & -8 & 5 & 0 \\ 5 & 0 & 4 & 2 \end{array}\right]$$

Given Matrix

(R1C2 indicates row 1, column 2)

$$\left[\begin{array}{ccc:c} 3 & 2.333 & 2 & 3 \\ 1 & -8 & 5 & 0 \\ 5 & 0 & 4 & 2 \end{array}\right]$$

R1C2 = 7/3 = 2.333
R1C3 = 6/3 = 2
R1C4 = 9/3 = 3

$$\left[\begin{array}{ccc:c} 3 & 2.333 & 2 & 3 \\ 1 & -10.333 & 5 & 0 \\ 5 & -11.656 & 4 & 2 \end{array}\right]$$

R2C2 = [-8 - (1 x 2.333)]
= $\underline{-10.333}$

R3C2 = [0 - (5 x 2.333)]
= $\underline{-11.656}$

Fig. 1-6. (cont.) Example Solution of a System of Linear Equations by Crout's Method.

$$\left[\begin{array}{cccc} 3 & 2.333 & 2 & 3 \\ 1 & -10.333 & -.2913 & +2913 \\ 5 & -11.656 & 4 & 2 \end{array}\right]$$

R2C3 = [5 - (1 x 2)]/(-10.333)
= -.2913

R2C4 = [0 - (1 x 3)]/(-10.333)
= +.2913

$$\left[\begin{array}{cccc} 3 & 2.33 & 2 & 3 \\ 1 & -10.333 & -.2913 & .2913 \\ 5 & -11.656 & -9.398 & 2 \end{array}\right]$$

R3C4 = [4 - (5 x 2) - (-11.65 x -.2913]
= 9.398

$$\left[\begin{array}{cccc} 3 & 2.33 & 2 & 3 \\ 1 & -10.333 & -.2913 & .2913 \\ 5 & -11.666 & -9.398 & 1.022 \end{array}\right]$$

The Auxiliary Matrix

R3C4 = [2 - (5 x 3) - (-11.66 x .2913)]/(-9.398)
= 1.022

c. Construction of the Auxiliary Matrix

$$X = \left[\begin{array}{c} FR1C1 \\ FR2C1 \\ FR3C1 \end{array}\right]$$

Form of Final Matrix

(FR1C1 indicates row 1, column 1 entry in the Final Matrix)

$$\left[\begin{array}{cccc} 3 & 2.33 & 2 & 3 \\ 1 & -10.333 & -.2913 & .2913 \\ 5 & -11.666 & -9.398 & 1.022 \end{array}\right]$$

The Auxiliary Matrix

(AR1C1 indicates row 1, column 1 entry in the Auxiliary Matrix)

Fig. 1-6. (cont.) Example Solution of a System of Linear Equations by Crout's Method.

Z = AR3C4 = 1.022

$$\begin{bmatrix} FR1C1 \\ FR2C1 \\ 1.044 \end{bmatrix}$$

First Solution Entry in the Final Matrix

Y = [AR2C4 - (AR2C3 x FR3C1)]

= [.2913 - (-.2913 x 1.022)]

= .589

$$\begin{bmatrix} FR1C1 \\ .589 \\ 1.022 \end{bmatrix}$$

Second Solution Entry in the Final Matrix

X = [AR1C4 - (AR1C2 x FR2C1) - (AR1C3 x FR3C1)]

= [3 - (2.333 x .589) - (2 x 1.022)]

= -.418

$$\begin{bmatrix} -.418 \\ .589 \\ 1.022 \end{bmatrix}$$

Complete Form of the Final Matrix

X = -.418

Y = .589

Z = 1.022

Solutions to the Original System of Equations

d. Construction of the Final Matrix

Fig. 1-6. (cont.) Example Solution of a System of Linear Equations by Crout's Method.

$$X - 8Y + 5Z = [(-.418) - 8(.589) + 5\ (1.044)]$$

$$= -.020$$

$$\simeq 0$$

e. Check of Solution by Substitution into Second Equation of the System

Fig. 1-6. Example Solution of a System of Linear Equations by Crout's Method.

CHAPTER 2

ECAP'S METHOD OF SOLUTION FOR TRANSIENTS

General Considerations

The response of an electrical network with linear, lumped, time-invariant elements and non-sinusoidal excitations can be described by a system of integro-differential equations. Exact solutions are possible, but more often than not they are difficult equations as well. If the elements are time variable and/or non-linear, exact solutions may be almost impossible. The response of a network, even with non-linear and time variable elements can be approximated by several different numerical techniques. ECAP converts a system of integro-differential equations into a system of recursion equations with time as an indexed variable. If the network response is known at time t_k; then the solution of the recursion equations yields the response at time $t_{k+1} = t_k + \Delta t$, where Δt is a small fixed increment of time. Time is indexed in steps of Δt from some prescribed initial time t_o to some final time t_f. If an element value is to change at a specific time, the recursion equations can be adjusted for this change before the next indexed time solution. Thus a time variable or non-linear element can be described by a piecewise linear approximation.

Numerical, time-indexed, approximate solutions are not without their own inherent difficulties. If the time step is too large, the solution may be useless due to approximation errors. If the time step is too small, the differences between

successive solutions may be too small for accurate calculations. Since the system of equations must be solved for each time step, short time steps require more computer time for solution. One technique for selecting an 'efficient' time step is to initially select a time step that is known to be too large; and allow the computer to approximate a solution. The time step can then be successively reduced by a factor of ten, the solution after each reduction being compared with preceding solution. When the difference between two successive solutions is adequately small, the final time step is 'efficient' enough. Only a few reductions of time step are necessary if the initial time step is reasonable. But even if it is grossly too large, the machine time used to select an 'efficient' time step will be shorter than the final running time; and for commercial use may be less expensive than paying an engineer to select a better initial time step.

Another technique for selecting an initial time step requires the calculation of 'isolated' time constants of pairs of elements connected to the same node. If a resistor and capacitor were connected to the same node (along with many other elements), the 'isolated' time constant would be the RC product. Only the time constants resulting from the smallest resistor and capacitor and the largest inductor connected to each node need be calculated. One tenth of the smallest time constant at any node can serve as a 'good' guess for an initial time step. This step for some circuits could waste some computer time; but if the

time constants of all the nodes are not too different, the guess won't be bad. The ECAP/1620 USER'S MANUAL (1) has several pages of discussion on time step selection.

ECAP will solve a transient analysis problem with any user specified time step that can be described in seconds within the bounds of the computer number system. Solutions are provided at integer multiples of the same time step, so it is usually more convenient for man and machine to agree upon integer decimal fractions of a second as a time step. It was mentioned earlier in this chapter that a time step could be too short for the accuracy of the calculations. Since ECAP/360 performs all calculations in double precision (using sixteen significant figures), this is not a severe problem.

The standard circuit branch for ECAP/360 Transient Analysis is the same as shown in Fig. 1-1 of Chapter 1, except that the branch element can be a resistor, inductor, or capacitor. The polarity markings are applicable as shown. ECAP/360 solves for node voltages, element currents, branch currents, branch voltages, element voltages, and instantaneous branch power loss. Instantaneous branch power loss is the product of element voltage and element current. It will not be zero in general for inductors and capacitors, as energy can be stored in them.

Development of the Recursion Equations for ECAP Transient Analysis

Conversion of the description of an electrical network from a system of integro-differential equations to a system of

recursion equations requires a systematic method to approximate derivatives and integrals. Since ECAP converts the circuit topology into nodal equations; the functions that must be approximated are, for capacitors:

$$J_k = C \left. \frac{d(e_m - e_n + E)}{dt} \right/_{t=t_k} \tag{8}$$

and for inductors:

$$J_k = \frac{1}{L} \int_{t_o}^{t_k} (e_m - e_n + E)\, dt + J_o \tag{9}$$

where:

t_o = initial time,

t_n = current time,

J_k = element current at time t_k

J_o = initial element current in the inductor,

C = capacitance of the branch in farads,

L = inductance of the branch in henrys,

e_m = voltage of the 'from' node of the branch,

e_n = voltage of the 'to' node of the branch,

E = voltage of the branch voltage source.

The recursion equation to approximate the derivative can be derived from the defining equation from calculus:

$$\frac{d\, f(t)}{dt} = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t}$$

by forcing Δt to remain at some constant small value instead of approaching zero. If the time step is defined as Δt, and the branch voltage source E is assumed to be zero: the approximation equation for current in a capacitor is:

$$J_k \simeq \frac{E_C}{\Delta t} [e_m(t_k+\Delta t) - e_n(t_k+\Delta t) - e_m(t_k) + e_n(t_k)] \qquad (10)$$

where:

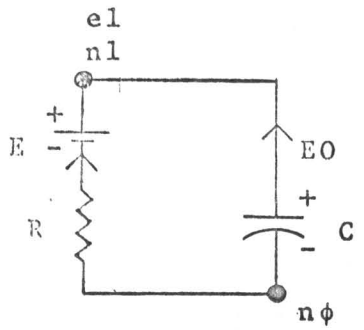
$e_m(t_k+\Delta t)$ = the 'from' node voltage at time $t_{k+1}=t_k+\Delta t$,

J_k = the current in the capacitor at some time between t_k and t_{k+1} (by the mean value theorem for derivatives).

In Eq.10 is the factor $C/\Delta t$ has the units of mhos, the same as conductance. If Eq. 3 is accepted as an equality, it can be used as a recursion equation. Figure 2-1 shows a simple RC circuit and its differential, Laplace, and recursion equations. The equations follow from Kirchoff's nodal law. The recursion equation of Fig. 2-1 can be rewritten with the unknown node voltages on the left hand side as:

$$-\left[\frac{1}{R} + \frac{C}{\Delta t}\right] e_1(t_{k+1}) = \left[\frac{-C}{\Delta t}\right] e_1(t_k) - \frac{E}{R} \qquad (11)$$

What happened to the initial voltage E_o on the capacitor in Figure 2-1 when the recursion equation was written? The obvious answer: it was included as $e_1(t_o)$. To obtain a solution for the initial element currents, ECAP replaces the capacitor with



EO is the initial voltage on the capacitor.

Differential Equation:

$$0 = e_1/R + C[de_1/dt] - E/R$$

Laplace Equation:

$$0 = [1/R + Cs]\ e_1(s) - E/R - C\ [EO]$$

Recursion Equation:

$$0 = [1/R]\ e_1(t_{k+1}) + [C/\Delta t][e_1(t_{k+1}) - e_1(t_k)] - E/R$$

Figure 2-1. RC Circuit and its Equations

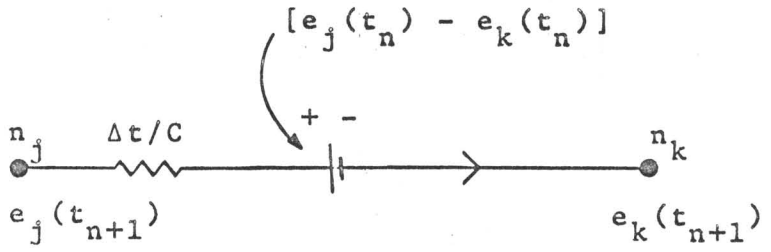


Figure 2-2. Transient Analysis Equivalent Circuit for a Branch Capacitor

a very small resistor (.001 ohms, unless specified otherwise by the program user) and executes a DC solution. If the capacitive branch had a time variable voltage source, the program user must specify the voltage waveform. ECAP approximates the voltage waveform as a piecewise linear function; therefore, the 'slope' of the voltage waveform is known, the 'slope' is approximately the same as dE/dt, the term that was assumed to be zero in the development of Eq. 10. Equation 10 could be easily adjusted to include dE/dt. Figure 2-2 shows an equivalent circuit for a capacitor that can be useful in writing recursion equations.

Recursion equations to approximate inductors in circuits can be developed from several different numerical integration techniques. ECAP divides the 'area under the function' to be integrated into a series of small rectangles. Integration is approximated by summing the area of the small rectangles. Figure 2-3 shows a time function broken down into the appropriate rectangles. The time interval between successive solution times, say between t_2 and t_1, is the time step Δt. Two rectangles are superimposed on each time interval. The two rectangles divide the time interval in the middle, so the base of each rectangle is $\Delta t/2$. It can be seen from Fig. 2-3 that the ordinates of the rectangles are determined by the value of the function at the beginning and end of the time interval. The sum of the 'area' of the two rectangles in a time interval is approximately the contribution of the function to the

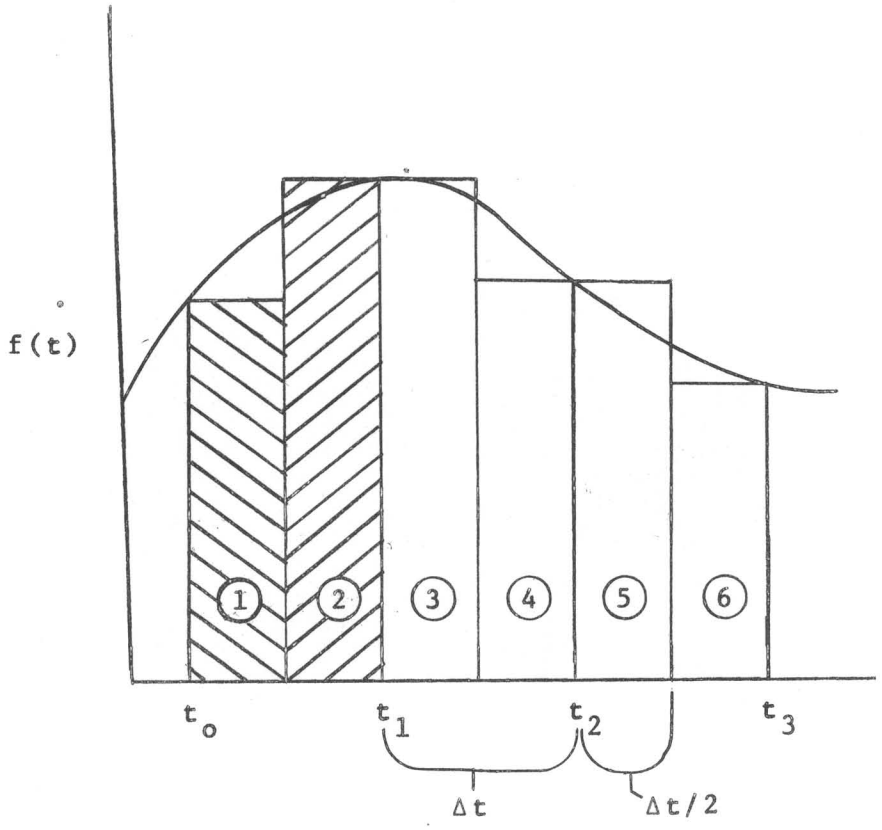


Figure 2-3. Approximation of an Integral

integral in that interval. Using the numbered rectangles of Fig. 2-3, the contribution to the integral in the time interval from t_2 to t_3 would be approximately the sum of the areas of rectangles numbered 5 and 6. One rectangle in the interval over-estimates the 'area under the curve'; the other rectangle in the interval under-estimates the 'area under the curve'. The sum of the two rectangles balances the under-estimate and the over-estimate. If the time step Δt (the length of the time interval) is small, the function is almost linear and the approximation of the integral in that interval is very good. The approximate value of the integral between an initial time t_o and current time t_k can be found by summing the rectangles of several time intervals. Several examples of approximating the integral of the function f(t), using A(s) as the area of rectangle number s, are:

$$\int_{t_o}^{t_1} f(t)\, dt = A(1) + A(2)$$

$$= [\tfrac{1}{2} f(t_o) + \tfrac{1}{2} f(t_1)]\, \Delta t$$

$$\int_{t_o}^{t_2} f(t)\, dt = A(1) + A(2) + A(3) + A(4)$$

$$= [\tfrac{1}{2} f(t_o) + f(t_1) + \tfrac{1}{2} f(t_2)]\, \Delta t$$

$$\int_{t_o}^{t_3} f(t)\,dt = A(1) + A(2) + A(3) + A(4) + A(5) + A(6)$$

$$= [\tfrac{1}{2}f(t_o) + f(t_1) + f(t_2) + \tfrac{1}{2}f(t_3)]\Delta t$$

$$\int_{t_o}^{t_{k+1}} f(t)\,dt = \tfrac{1}{2}f(t_o)\Delta t + \sum_{j=1}^{k-1} f(t_j)\Delta t \qquad (12)$$

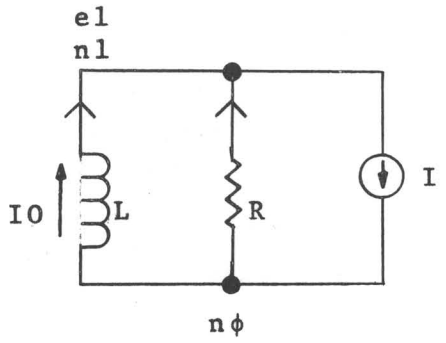
$$+ \tfrac{1}{2}[f(t_k) + f(t_k) + f(t_{k+1})]\Delta t$$

$$= \int_{t_o}^{t_k} f(t)\,dt + \tfrac{1}{2}[f(t_k) + f(t_{k+1})]\Delta t$$

Equation 13 is the application of Eq. 12 to approximate the integral of Eq. 9.

$$J_{k+1} = J_k + \tfrac{1}{2}(\Delta t/L)[e_m(t_{k+1}) - e_n(t_{k+1}) - e_m(t_k) + e_n(t_k)] \qquad (13)$$

Figure 2-4 shows Eq. 13 applied to a simple RL circuit. All the terms in the recursion equation of Fig. 2-4 are known except the node voltage $e_1(t_{k+1})$, so the equation 'predicts' that term. What happened to the initial inductor current in the recursion equation? The obvious answer: it was included at the initial time t_o in the term J_o. Figure 2-5 shows an equivalent circuit for an inductor that can be useful in writing recursion equations. Since the term $\Delta t/2L$ has the units of mhos, the use of the resistor in the equivalent circuit is justified. In the recursion formulas for both inductors and capacitors, the time step Δt plays the role of 'frequency'.



IO is the initial current in the inductor

Differential Equation:

$$0 = e_1/R + [1/L]\int_{t_o}^{t} e_1\, dt - I$$

Laplace Equation:

$$0 = [1/R + 1/Ls]\, e_1(s) + [IO/s] - I(s)$$

Recursion Equation:

$$0 = 1/R + [\Delta t/2L][e_1(t_{k+1}) - e_1(t_k)] + J_k - I_{k+1}$$

Figure 2-4. RL Circuit and its Equations

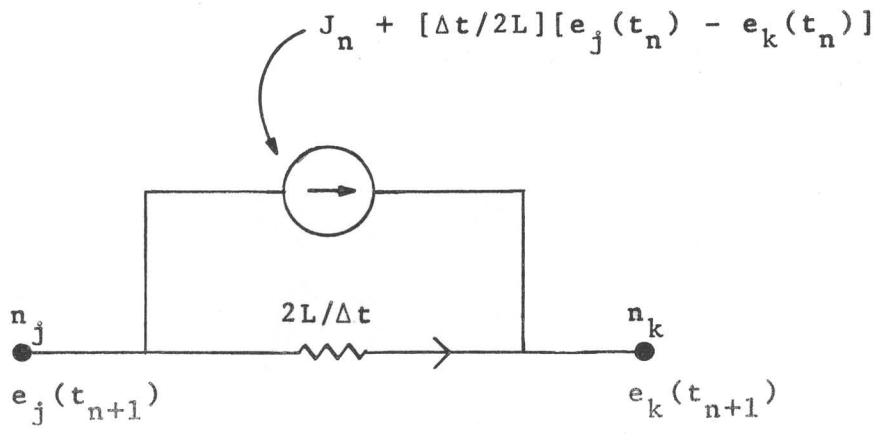
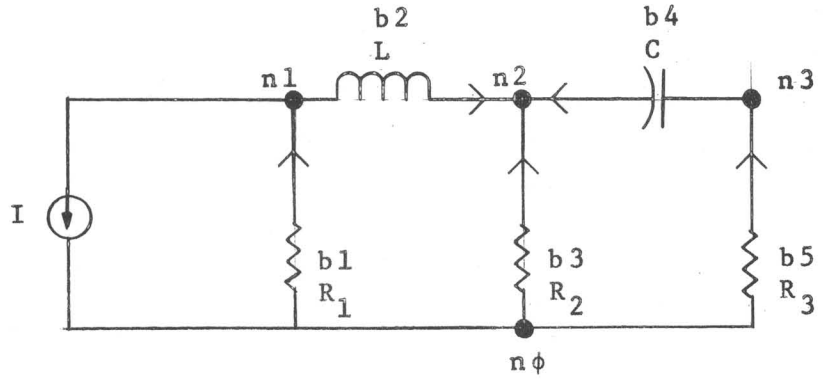


Figure 2-5. Transient Analysis Equivalent Circuit for a Branch Inductor

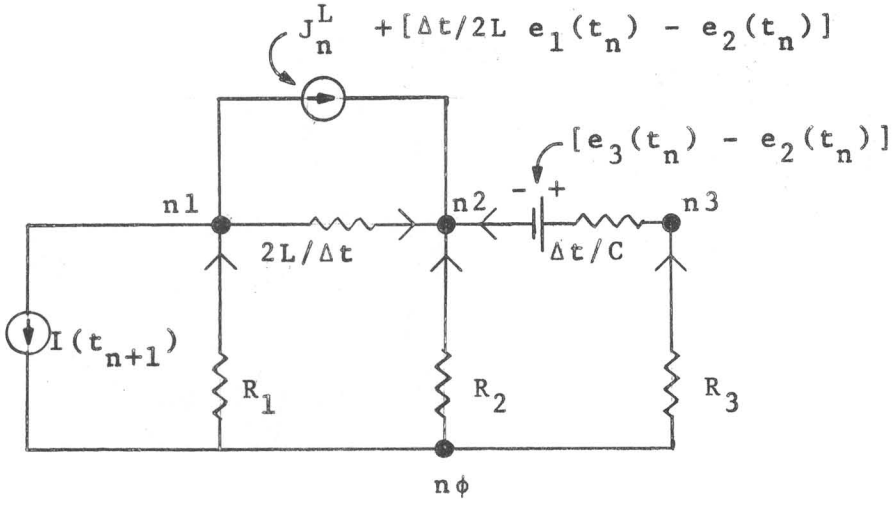
Once the circuit configuration has been specified, the circuit can be converted into a transient analysis equivalent circuit and the recursion equations written by inspection. Figure 2-6 shows a T circuit converted into an equivalent circuit. The term J_n^L is used to denote the current in the inductor at time t_n. The circuit recursion equations are shown in matrix form in Fig. 2-7. It is apparent from Fig. 2-7 that a system of rules for writing the recursion equations can be developed that are almost identical to the rules used to write the nodal equations in DC and AC analysis. Using the same notation as was used in DC Analysis in Chapter 1, the matrix equation can be written as:

$$[ZPRL][E] = [EQUCUR]$$

The matrix equation can be solved for [E], and the result is the node voltage at time t_{n+1} in terms of circuit voltages at time t_n. The results of one solution are substituted into [EQUCUR] with the time indexed by one and the equation solved again. If the values of the elements remain constant during the interval of the solution, the matrix [ZPRL] can be inverted and used to shorten the number of calculations necessary. Each time an element changes value, a new inverse is calculated. ECAP inverts [ZPRL] with a method that combines Gaussian elimination and Crout's method (11). An 'identity' matrix of the same rank as [ZPRL] is placed 'to the right' of [ZPRL], and the inversion process proceeds in exactly the same manner that was described in Chapter 1.



a. Actual Circuit



b. Equivalent Circuit

Figure 2-6. Example T Circuit and its Transient Analysis Equivalent

$$\begin{bmatrix} (1/R_1 + \Delta t/2L) & (-\Delta t/2L) & 0 \\ (-\Delta t/2L) & (1/R_2 + t/2L + C/\Delta t) & (-C/\Delta t) \\ 0 & (-C/\Delta t) & (1/R_3 + C/\Delta t) \end{bmatrix} \begin{bmatrix} e_1(t_{n+1}) \\ e_2(t_{n+1}) \\ e_3(t_{n+1}) \end{bmatrix}$$

$$= \begin{bmatrix} J_n^L + (\Delta t/2L)(e_1(t_n) - e_2(t_n)) + I(t_{n+1}) \\ -J_n^L - (\Delta t/2L)\, e_1(t_n) + (\Delta t/2L - C/\Delta t)\, e_2(t_n) + (C/\Delta t) e_3(t_n) \\ (C/\Delta t)\,(e_2(t_n) - e_3(t_n)) \end{bmatrix}$$

Figure 2-7. Matrix Recursion Equation of the Example T Circuit of Fig. 2-6

Some Important Differences Between ECAP/360 and ECAP/1620 Transient Analysis

Earlier in this chapter, it was mentioned that selection of a time step that was much too short could influence the accuracy of the solution. Since ECAP/360 calculates its solutions using sixteen significant figures this problem was not important. ECAP/1620 uses only eight significant figures, therefore time step selection can be critical.

ECAP/1620 will allow a program user who is seated at the console to interrupt the solution, change circuit element values, and restart the solution at the initial time. The user can also specify either printed or punched output. Also, if the time computer time available to solve the problem completely exceeds the time available; the user can request that enough data be punched on cards to restart the solution at a later time without repeating the solutions already obtained. ECAP/1620 will solve for element currents and node voltages only.

ECAP/360 will solve for node voltages, branch voltages, branch currents, element voltages, element currents, and instantaneous branch power loss. The capability to interrupt and re-start a solution at a later time is not needed with ECAP/360 because of the improved speed of calculation of IBM's System 360 over the IBM 1620. Since most ECAP/360 users will not be allowed access to the console of the computer, the ability to change circuit elements and start the solution again at the initial time from the console could not be used if it were available.

Appendix A is a duplication of an IBM Technical Letter which discusses some errors in the method of calculation of ECAP/1620. The handwritten notes on the right hand margin of Appendix A indicate whether the error applies to ECAP/1620 alone, or both ECAP programs. The notes are "as received" from the author of ECAP/360, Mr. Gerald M. Hogsett; an IBM employee.

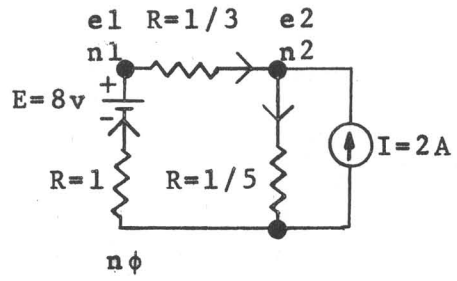
CHAPTER 3

INSTRUCTION GUIDE FOR SOLVING DC CIRCUITS WITH ECAP/360

Basic Considerations

ECAP/360 describes a DC circuit with a set of equations derived from Kirchoff's nodal law: 'the sum of currents into a node must be zero'. The set of nodal equations is converted into a matrix equation of the form [Y][E] = [I]. [Y] is called the nodal conductance matrix and [I] is called the equivalent current vector. Figure 3-1 shows a simple circuit and its associated equations. The solution of the set of nodal equations will yield the node voltage matrix, [E]. ECAP solves the matrix equation by premultiplying both sides of the equation by the inverse of [Y]. The inverse is formed from [Y] by Gaussian elimination. Figure 3-2 shows an example of Gaussian elimination inversion. The currents in the elements can be found easily once the node voltages are known by applying Ohm's law. The writing of the equations and the application of Ohm's law tacitly assumes that a positive direction of current flow and the node connections have been specified, along with positive conventions for voltage and current sources. While an individual might 'cut some corners' in specifying these conventions if he were writing the circuit equations himself; ECAP requires a complete and accurate description of assigned conventions. To simplify the problem of specifying positive conventions, a 'Standard Circuit Branch' has been devised. Figure

Circuit



Node Equations

$$4\,e_1 - 3e_2 = 8/1 = 8$$

$$-3\,e_1 + 8e_2 = 2$$

Matrix Equation
[Y][E] = [I]

$$\begin{bmatrix} 4 & -3 \\ -3 & 8 \end{bmatrix} \begin{bmatrix} e1 \\ e2 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \end{bmatrix}$$

Figure 3-1. A Simple DC Circuit and its Equations

Initial [Y] matrix

$$\begin{bmatrix} 4 & -3 \\ -3 & 8 \end{bmatrix}$$

Augmented matrix

$$\begin{bmatrix} 4 & -3 & : & 1 & 0 \\ -3 & 8 & : & 0 & 1 \end{bmatrix}$$

First Step

$$\begin{bmatrix} 1 & -.75 & .25 & 0 \\ -3 & 8 & 0 & 1 \end{bmatrix}$$

Second Step

$$\begin{bmatrix} 1 & -.75 & .25 & 0 \\ 0 & 5.75 & .75 & 1 \end{bmatrix}$$

Third Step

$$\begin{bmatrix} 1 & -.75 & .25 & 0 \\ 0 & 1 & .1302 & .1740 \end{bmatrix}$$

Last step, inverse on right hand side.

$$\begin{bmatrix} 1 & 0 & .3477 & .1302 \\ 0 & 1 & .1302 & .1740 \end{bmatrix}$$

Figure 3-2. Matrix Inversion by Gaussian Elimination

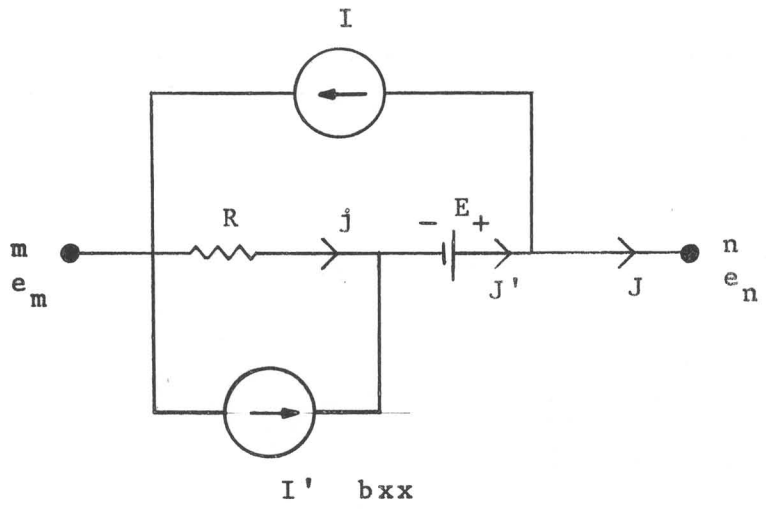


Figure 3-3. ECAP DC ANALYSIS 'Standard Circuit Branch'

3-3 shows the DC analysis 'Standard Circuit Branch' with its associated conventions. The definitions for the items shown in Figure 3-3 are:

R is the branch element resistance,

E is the branch fixed-magnitude voltage source,

I is the branch fixed-magnitude current source,

bxx indicates branch number xx,

j is the current flow in the resistor,

I' is the branch dependent current source (which is controlled by the current flow in the resistor of another branch),

J' is defined as the element current,

J is defined as the branch current,

m is the 'from' node number of the branch,

n is the 'to' node number of the branch,

e_m is the node voltage at node m,

e_n is the node voltage at node n.

All positive conventions are shown in Figure 3-3 with arrows or polarity (+ & -) marks. Positive element and branch currents flow from the 'from' node of the branch. Once the positive direction of current in a branch resistance is specified by the user; all other conventions for that branch are implied by the 'Standard Circuit Branch'. Every branch must have a non-zero resistance, so every voltage and current source must have an associated resistance (though it can be extremely large or small). If a source is not wanted in the branch, the value of that source becomes zero. For example; if a voltage source is not wanted

in a branch, that branch E = 0. ECAP assumes that every source in a branch is zero unless it is specified otherwise by the program user. If a source is desired that has a polarity or current flow opposite to the positive convention, it is specified with a negative magnitude.

Every node and branch of the circuit must be uniquely numbered. One node must be assigned as the zero-voltage (reference) node. The reference node is assigned the number zero. All node numbers from zero to the highest node number must be assigned. If there are fifteen nodes in a circuit; the highest node number is fourteen, and all numbers between zero and fourteen must be assigned. No node or group of nodes can be isolated (i.e.; without some connection to the reference node). Branch numbers must run from one to the highest branch number. Branches and node numbers are arbitrarily assigned by the program user, as are positive current directions. Figure 3-4 shows a circuit with all the conventions assignments that would be needed to describe the circuit for ECAP. Notice that the 'Standard Circuit Branch' need not be completely shown for each branch, only the items physically present in the network need be shown.

All the labeling and assigning of conventions to a circuit may seem to be a big nuisance, but it is not nearly as bad as trying to solve a circuit with many nodes and branches by hand. ECAP/360 will handle circuits with up to:

50 nodes, not including the reference node,

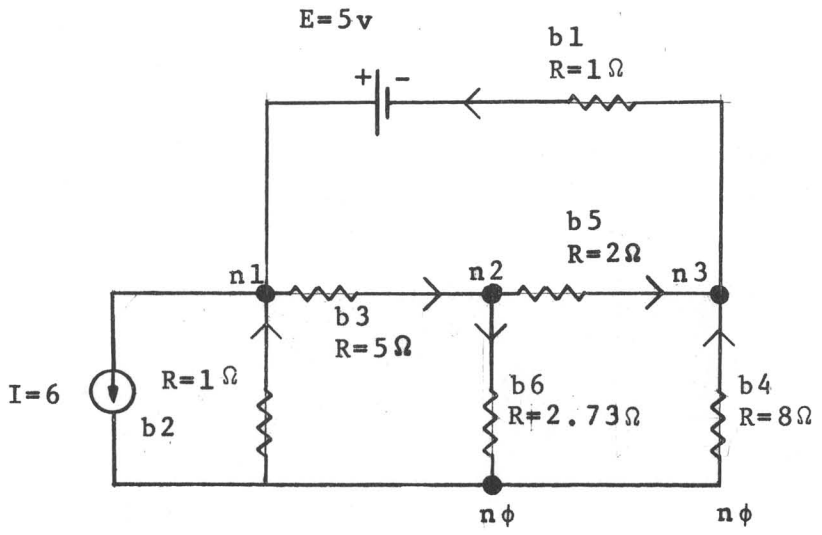


Figure 3-4. Example Circuit with Conventions Shown

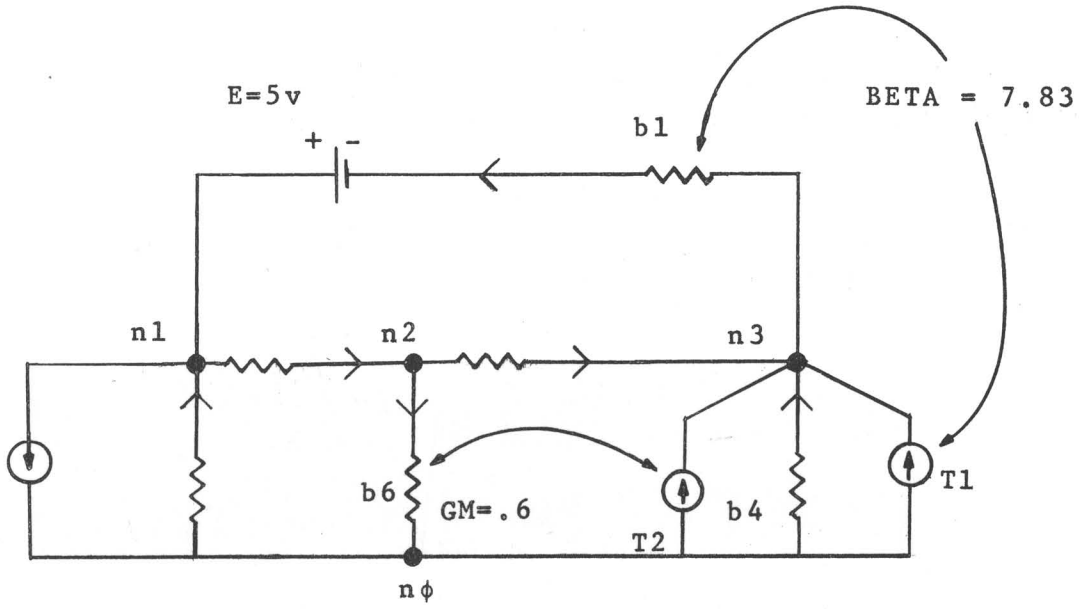


Figure 3-5. Example Circuit of Figure 3-4 with Two Dependent Sources Added

200 branches,

200 fixed-magnitude voltage sources,

200 fixed-magnitude current sources,

200 dependent current sources,

NO dependent voltage sources.

Even with 'easy' numbers, a circuit with fifty independent nodal equations would be enough to discourage any 'paper and pencil' solutions. If a dependent voltage source is needed, the dependent voltage source can be converted into a Norton equivalent dependent current source by the program user.

A dependent source is really an isolating current amplifier. It is physically located in its 'to' (controlled) branch, but its current flow is regulated by the current flow in the resistor of some other branch. The regulating (controlling) branch is the 'from' branch of the dependent source. The link between the 'to' and 'from' branch can be specified by either a current gain (BETA) or a transconductance (GM). When a transconductance is specified, the voltage across the resistor of the 'from' branch controls the source. Notice that if the 'from' branch has an internal voltage source, the voltage across the resistor is not the same as the voltage across the branch. The voltage across the resistor is its IR drop. For example: if the voltage across the 'from' branch is 5 volts, its voltage source +1 volt, and the transconductance between the 'from' and 'to' branches is 3.1 mhos; then the current in the dependent current source in the 'to' branch is: (3.1)(5+1) = 18.6 amperes. BETA and GM are related by the formula

$$GM = BETA/R_f$$

where R_f is the resistance of the 'from' branch. Figure 3-5 shows the circuit of Figure 3-4 with the addition of two dependent sources. Some of the branch labels have been deleted. Notice that a branch may have more than one dependent current source. It is <u>not</u> possible for a branch to have more than one fixed-magnitude current source, or more than one fixed-magnitude voltage source. Dependent voltage sources are not allowed.

Table 3-1 shows the definition of several parameter. Table 3-1 assumes that branch z has 'from' node x, 'to' node y; and the dependent source in branch z has 'from' branch q and transconductance GM_{qz}. In Table 3-1, it is important to note the definition of branch current, element current, and branch power loss. Specifically, it is the method of calculating branch power loss that justifies the Figure 3-3 representation of the 'Standard Circuit Branch'. The 'Standard Circuit Branch' of Figure 3-3 (which applies to ECAP/360) is not the same as the one shown in the IBM ECAP/1620 USER'S MANUAL; though this manual in general applies to ECAP/360 also.

Basic Instructions Required to Use ECAP/360 DC Analysis

ECAP/360 at the present is available from disc storage at the KSU Computing Center. Instructions to the computer can be broken into eight classifications:

Parameter	Symbol	Defining Equation	Available as an ECAP/360 Output	Output 'PRINT' Code
Node x Voltage	e_x	Solution of Eq.'s	YES	NV
Branch z Voltage Source	E_z	User Data	NO	
Branch z voltage	V_z	$V_z=e_x-e_y$	YES	BV
Element Voltage	V'_z	$V'_z=V_z+E_z$	YES	CV
Branch Current Source	I_z	User Data	NO	
Branch z Dependent Source	I'_z	$I'_z=GM_{qz}V_q$	NO	
Branch z Resistance	R_z	User Data	NO	
Resistor Current	j_z	$j_z=V'_z/R_z$	NO	
Element Current	J'_z	$J'_z=j_z+I'_z$	YES	CA
Branch z Power Loss	P_z	$P_z=J'_z\ V'_z$	YES	BP
Branch z Current	J_z	$J_z=J'_z-I_z$	YES	BA

Table 3-1. ECAP DC ANALYSIS definitions for Parameters.

1. Commands to get the program from storage
2. User circuit data,
3. Solution Controls,
4. Output Specifications,
5. Commands,
6. System Controls,
7. Comments,
8. Termination of job instructions.

All input to the computer is by punched cards, and all output from the computer is printed. There is no punched card output option. In the remainder of these instructions; the standard eighty entry data card will be represented by a /1 above the position where column one would be, and a 7 above column seven. Imbedded blanks are assumed to be one character entry long unless a column number entry is shown above the character following the blanks. No entrys are allowed in columns 73-80. A sample of two comment cards (one after the other) by this system of representation would be:

```
/1      7
 C      D.C. SPECIAL TEST PROGRAM
 C      UNBALANCED EDISON 3-WIRE CIRCUIT
```

There are five special cards required to get the program from the disc. They are <u>not</u> part of ECAP, but are control cards required by the Computing Center for orderly operation. All five of them <u>must</u> be punched in the EBCDIC (extended binary code) character set. The EBCDIC characters are the ones punched by an IBM 29 Card Punch, or multiple punched on an IBM 26 Printing Card Punch. The 26 Printing Card Punch normally punches

BCD (binary code) characters. The first card must be a 'job card', exactly the same as for other programs submitted to the Computing Center. At present, this card is completely specified in KSU Computing Center Notice #52, dated 27 February 1968. An abbreviated form of this card which 'will work' is:

```
                1   1
/1              2   6
 //jobname  JOB (pano,time,lines),yername,MSGLEVEL=1
```

where

// must appear in columns 1 and 2,

jobname is 1-8 characters which name the job, the first of which must be alphabetic,

JOB must start in column 12,

(must appear in column 16,

pano is your job charge number,

time is the maximum running time in integer minutes,

lines is the maximum number of lines of printed output in integer thousands of lines,

yername is your own name with no imbedded blanks.

A sample of a job card would be (with a FALSE job number):

```
            1   1
/1          2   6
 //ALLERROR JOB (06W190978Q001,5,4),WWJONES,MSGLEVEL=1
```

If the running time of the problem exceeds the 'time' entry on the job card; the computer will automatically terminate the job. The job will also be terminated if the number of 'lines' is exceeded. Either the 'time' or 'lines' entry (or both) can be omitted. If 'time' is omitted, the data group inside the parentheses becomes (pano,,lines). There can be no blanks between the commas. The computer will assume a 'time' entry of

1 minute. If the 'lines' entry is omitted, the data group becomes (pano,time). The computer will assume a 'lines' entry of 1 (for 1000 lines). If both 'time' and 'lines' are omitted; the data group becomes (pano), and the assumed entrys are the same.

The remainder of the five special command cards as presently specified are:

```
/1     7
 //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
 //FT05F001 DD *
```

These five special cards must be the first five cards in any ECAP problem deck, and they must be in the order presented.

Two special cards are used to terminate an ECAP job, and they are the last two cards in an ECAP problem deck. In proper order, they are:

```
/1     7
       END
 /*
```

Comment cards can be placed anywhere after the first five cards and before the last two cards of the problem deck. Column 1 of a comment card must have the character C punched in it. Two examples of comment cards were included as an earlier example.

The remaining cards to be discussed will be assumed to be punched in EBCDIC, but a method for using BCD cards will be shown later.

Two types of data cards are used to describe the problem circuit to ECAP. The first type specifies all the data from one

branch, except the dependent source data. This card is called a 'branch' or 'B' card. The standard form of a 'B' card is:

```
/1      7
 Bxx    N(ww,zz),R=rr,E=ee,I=ii
```

where

B indicates that it is a 'B' card,

xx is the number of the branch, and can appear in any of the columns 2-5,

N indicates the node connections of the branch,

ww is the 'from' node of the branch,

zz is the 'to' node of the branch,

rr is the resistance of the branch in ohms,

ee is the voltage of the fixed-magnitude voltage source in the branch,

ii is the current output of the fixed-magnitude current source in the branch,

If the conductance of a branch is known, the entry R=rr in the 'B' card could be replaced with G=gg. In this case, gg is in mhos.

The second type of data card specifies dependent source data. It is called a 'T' card after its column one entry. The standard form of a 'T' card is

```
/1      7
 Txx    B(bb,dd),BETA=tt
```

or

```
/1      7
 Txx    B(bb,dd),GM=kk
```

where

T indicates a dependent source ('T') card,

xx is the number of the dependent source,

bb is the 'from' branch of the dependent source,

dd is the 'to' branch of the dependent source,

tt is the current gain of the dependent source,

kk is the transconductance of the dependent source.

Only one 'T' card is needed for each dependent source, but the user has the option of specifying either BETA or GM. The dependent sources are uniquely numbered in order from one to the highest number. No number between one and the highest can be missing.

The node and branch numbers that appear on the data cards must be non-negative integers. All other data entrys can be up to eight digits, with or without a decimal, and with or without an E followed by an integer that indicates an exponential power of ten. The largest number accepted by ECAP is 7E75, which is 7 followed by 75 zeros. The exponent can also be a negative integer. Sometimes the program output uses D instead of E, but it means the same thing. Using D on an input data card to indicate an exponent is not allowed. The following examples show the same number represented in several equally valid forms:

```
-28.3
-28.300000
-283E-.1
-.283E2
-28300000E-6
```

The following examples are <u>not</u> valid numbers for ECAP:

```
+28.375943675844657     (too many digits)
-2E+9997                (exponent too large)
```

Several examples of 'B' and 'T' cards can be developed from the preceding discussion. Note that not all entrys are necessary on a 'B' card if either the E or I source are missing in the branch.

```
/1        7
 B7       N(0,50),R=5,I=1.076E-2
 B 16     N(17,6), R = 1 E -6, E=17, I =-1.2
 B198     N(1 ,4  ) R =        1000
 B  1     N(5,22),R=-16,E=12
 C
 T  8     B(8,7),GM=1.5
 T  5     B(2,6),BETA = .122 E +01
```

It can be seen from the preceding examples that the entrys can have a lot of imbedded blanks (an imbedded blank is a blank entry between two other characters) and be rather free in order. The entrys in column one must be either 'B' or 'T' for data cards, but never a blank. Putting the N or B in column seven is 'good form', but it can occur later than column seven but not before. R,E, and I entrys can occur in any order after the N(xx,yy) entry on a 'B' card, but it is 'good form' to adopt a uniform order for neat looking problems. ECAP produces the input data in its output along with its solutions, so a messy input form will give the same mess as an output. The solution output from ECAP will be in the same impeccable, neat form; whatever the input may be.

Sometimes the columns 7 - 72 may not provide enough space to completely specify all the information on a 'B' card. The data for that card can be continued on a second (or even third) card by placing an * in column six, and punching the remaining data in columns 7 - 72 of the continuation card. The continuation

card for a 'B' card must immediately follow that 'B' card in the problem deck. Since no entry is included in columns 1-5 to indicate which branch is being continued, the continuation card should be marked with a pen or pencil to show the branch number. An example of a 'B' card and two continuations would be:

```
/1      67
 B12     N(3,4), R = .12345678  E + 02  ,
         *       I = .23456789  E + 1   ,
         *       E = .87654321  E - 01
```

'T' cards can not be continued.

Solution Output Specification cards have the word PRINT starting in column seven followed by letter codes that control solution outputs. Only one PRINT card is needed for each problem. The letter code for output specification is shown in Table 3-2.

Code	ECAP Solution Output
NV	NODE VOLTAGES
BV	BRANCH VOLTAGES
CV	ELEMENT VOLTAGES
CA	ELEMENT CURRENTS
BA	BRANCH CURRENTS
BP	BRANCH POWER LOSSES

Table 3-2. Solution Output Codes.

Some typical PRINT card examples would be:

```
/1      7
        PRINT , NV , BA
        PRINT,NV,BV,BP
        PRINT,NV
```

```
PRINT,NV,BV,CV,CA,BA,BP
PRINT,   BP
```

Two command cards not previously mentioned are needed before a complete problem input deck can be described. The first command must appear in the deck just ahead of the data and specifies to ECAP that the problem to be solved is a DC circuit. This card is:

```
/1      7
        DC ANALYSIS
```

The second command card needed is the last card in the problem deck (but still before the END and /* cards). It commands the program to

```
/1      7
        EXECUTE
```

the solution.

Before considering an example of a complete problem deck, it should be mentioned that ECAP problems can be 'stacked' in batches on one job card. Instead of placing the END and /* cards after the EXECUTE card of the first problem, another DC ANALYSIS card followed by its data and EXECUTE card is included instead. Several problems can be stacked, and the END and /* cards are placed after the EXECUTE card of the last problem. All the problems are solved separately and the outputs are identified by also reproducing the input data along with the solutions. Some time is saved by not getting the ECAP program from disc storage for each solution. The number of problems that can be stacked and solved on one job card is limited only by the 'time' and 'lines' entrys on the job card. Figure 3-6

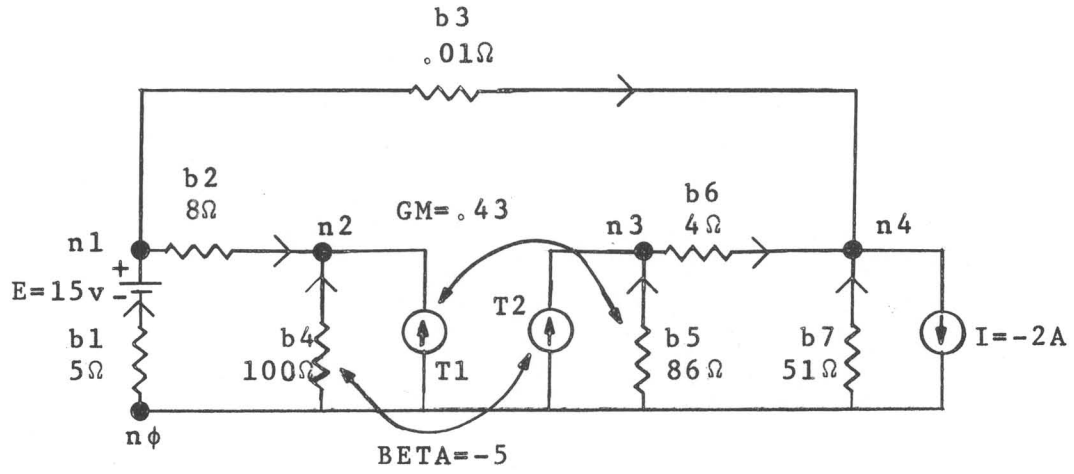
shows an example of converting a circuit to a problem input deck. The card classifications and card formats are reproduced in Table 3-3. Figure 3-6 includes the card classification number of each card for quick reference to Table 3-3.

ECAP has a built in check on the accuracy of its DC solutions. Since the current at each node should sum to zero, the program computes those sums. In most cases the sum will not be exactly zero since ECAP is limited to sixteen significant figure accuracy (eight in input and output) in its calculations. After computing the difference from zero at each node, ECAP sums the absolute value of these differences at all nodes. This sum is called the 'Current Unbalance' of the solution. Though the current unbalance is never printed out, if it is larger than .001 ampere an error message will be printed along with the solution obtained. The error message is: 'SOLUTION NOT OBTAINED TO DESIRED TOLERANCE'. This error usually occurs when the resistances or sources have an extremely large range within the same problem. The element values can be different by several powers of ten before solution accuracy is significantly influenced.

Sometimes it is inconvenient to punch cards with EBCDIC characters. If a group of cards, other than the first five cards in the deck, have been punched with BCD characters; they can be used if the card

```
/1       7
 *BCD
```

is inserted in the deck immediately ahead of the BCD punched



Card Classification Number

Program Cards

```
      /1       7
 1     //EXAMPLE  JOB (06W1909789Q001,2,2),JONES,MSGLEVEL=1
 1     //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
 1     //STEP1 EXEC PGM=ECAP
 1     //FT06F001 DD SYSOUT=A,DCB=RECFM=UA
 1     //FT05F001 DD *
 7     C
 7     C
 7     C       EXAMPLE DC ANALYSIS PROGRAM
 7     C
 5             DC ANALYSIS
 7     C
 2     B1      N(0,1), R = 5.0 , E = .15 E +02
 2     B2      N(1,2), R = 8
 2     B    3  N(1,4), G = .10 E -01
 2     B4          N(0,2), R = 100
 2     B5      N(0,3), R = 86
 2     B6      N(3,4),    R = 4.0
 2     B7      N(0,4) , R = .51000000 E +02
 2            *  I =  -.2000 E +01
 2     T 1     B(5,4), GM = .43
 2     T2      B(4,5), BETA = -5
 4             PRINT, NV , CA , BP
 5             EXECUTE
 8             END
 8     /*
```

Total Execution Time for this problem was: 0.31 minutes. Total Amount of Printed Output was: 8 pages, including the pages due to the HASP Log System of the KSU Computing Center.

Fig. 3-6. Example Circuit and Problem Deck for ECAP/360 DC ANALYSIS

Card Class. Number	Card Class.	Card Name	Card Format
			/1 7
1	Commands to get the Program	jobcard	(as currently specified)
		None	//JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
		None	//STEP1 EXEC PGM=ECAP
		None	//FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
		None	//FT05F001 DD *
2	User's Circuit Data	'B'	Bxx N(y,z), R=r
		'B'	Bxx N(y,z), R=r , E=e , I=i
		Contin-uation	* remaining data of a 'B' card
		'T'	Tqq B(a,b), GM = gg
		'T'	Tqq B(a,b), BETA = mm
4	Output Spec.	'Print'	PRINT,NV
		'Print'	PRINT,NV,BV,CV,CA,BA,BP
5	Commands	None	DC ANALYSIS
		Execute	EXECUTE
6	System Control	None	*BCD
		None	*EBCDIC
7	Comments	Comment	C any comment necessary to
		Comment	C understand the problem
8	End of job	End	END
	Instruc-tions	None	/*

Table 3-3. ECAP/360 DC ANALYSIS Input Cards

group of cards. If later in the deck EBCDIC punched cards are included (say in another problem), the card

```
/1      7
 *EBCDIC
```

converts the reading interpretation back to EBCDIC characters. In the absence of instructions, ECAP assumes the cards are all EBCDIC. If this assumption is wrong, a battery of error messages will be produced. The error messages will indicate by card number (in the order it appears in the deck) the card (cards) in error and also pinpoint the column of that card where an incorrect character occurs. If say a PRINT or EXECUTE card starts its first letter in column one instead of column seven, the program will pinpoint the error.

It was mentioned earlier in this chapter that an ECAP program also exists for the IBM 1620. Since ECAP/360 is a greatly expanded version of ECAP/1620, persons specifically interested in using ECAP 1620 should become familiar with both the ECAP 1620 USER'S MANUAL and the ECAP/1620 SYSTEM MANUAL. An object deck for ECAP/1620, along with all the ECAP/1620 manuals, are available at the operations office of the KSU Industrial Engineering Department's IBM 1620.

Additional ECAP DC ANALYSIS Solution Features

It was mentioned earlier that ECAP solves DC problems by forming the matrix equation [Y][E] = [I] and inverting [Y]. If the output specification code MI is included in the PRINT card, the program will print out the matrices [Y], $[Y]^{-1}$, and [I]. [Y] will be called the 'NODAL CONDUCTANCE MATRIX' in the output, $[Y]^{-1}$ will be called the 'NODAL IMPEDANCE MATRIX', and [I] will be called the 'EQUIVALENT CURRENT VECTOR'. An example output would be:

```
NODAL CONDUCTANCE MATRIX

ROW NO.    COLS.
  1        1 - 2    .49874361E-01   .00000000E-99
  2        1 - 2    .13562886E-02   .76742981E+02

EQUIVALENT CURRENT VECTOR

NODE NO.   CURRENT
  1        .47385299D-02
  2        .85546231D+01

NODAL IMPEDANCE MATRIX

ROW NO.    COLS.
  1        1 - 2    .76649328E+02   .49573876E-04
  2        1 - 2    .12345678E+01   .23456789E-02
```

The preceding example is not the result of an actual calculation, but is included only to show the form of an MI output. Row one of the conductance matrix and the current vector corresponds to the terms in a nodal equation written at node one. Likewise, row n would be an equation written at node n.

Sometimes a user may want to repeat a solution with only a few values changed. It would be inconvenient to punch a whole new deck of cards for the 'new' problem. A short method consists of placing the card

```
/1      7
        MODIFY
```

after the EXECUTE card of the 'old' problem; and then new 'B'

or 'T' cards in an abbreviated form for those branches and dependent sources that are to be changed. Only the value of elements can be changed. Node connections or the number of nodes cannot be changed with the MODIFY procedure. If the resistance in branch 2 was to be changed to 4 ohms from 8 ohms, the card:

```
/1      7
 B2     R = 4
```

would be included after the MODIFY card. Other examples of abbreviated data cards would be:

```
/1      7
 B 3    R = 7 , I = 12
 B 7    R = 1E+01 , E = -17
 B 6    E = 0.0
 T 1    BETA = 6
 T5     GM = .001
```

It is seen that only the 'B' or 'T' card to be modified is included after MODIFY, and in its abbreviated form only the number and the element with the changed value needs to be entered. A new EXECUTE card is placed after the last of the modified cata cards, and a new solution is produced. No new PRINT card is needed, as the 'old' PRINT card carrys over to the MODIFY solution. The MODIFY solution is a completely new solution and not an approximation from the 'old' solution; therefore the problem can be modified several times without a deterioration in accuracy. Each successive modification must have its own MODIFY, data, and EXECUTE cards. The END and /* cards are placed after the EXECUTE of the last MODIFY solution to terminate the job. There is one limitation on MODIFY solutions; not more than 50 parameters (R,L,C,E,I, and GM) can be changed at the same time.

Sometimes solutions are needed when only one element value is varied over a range of values. Using the MODIFY method would require several cards; but again a shorter method exists. The element value can be specified using a data group of the form:

p1 (p2) p3

where

p1 is the initial value of the element

p3 is the final value of the element

The meaning of p2 is best described by an example. Given that the entry in a data card is R = 5 (3) 20; the first solution would use R = 5, and then three additional solutions would be produced with R = 10, R = 15, R = 20 respectively. Thus the increments of a value is seen to be (p3 - p1)/p2, and p2 determines the number of equal steps from the initial to the final value. Both p1 and p3 can be any valid number, but p2 must be a positive integer. The variation of one element value can be included in a MODIFY solution as long as only one element in the MODIFY solution has a variation specified. An example that shows a variation and modification is:

```
/1      7
        DC ANALYSIS
 C      INITIAL PROBLEM
 B1     N(0,1), R = 10 (9) 100
 B2     N(1,2), R = 5, I = -6
 B3     N(2,0), G = 8
 T1     B(1,3),BETA = 5
        PRINT, NV
        EXECUTE
        MODIFY
 C      FIRST MODIFICATION AND VARIATION
 B1     R = 50
 T1     BETA = 10 (6) 80
        EXECUTE
        MODIFY
```

```
C       SECOND MODIFICATION AND VARIATION
B2      I = -10 (5)+10
        EXECUTE
        END
/*
```

The solutions produced by a variation will include a print out of the element value along with its corresponding solution. This prevents the user from getting lost in a flood of solutions.

Once a DC circuit with exact element values has been solved, the designer must face the reality that exact value elements are not available. If any element value is different from the exact value specified in the solution, all node voltages will be different too. The partial derivative of node voltage with respect to an element's value (with all other element values held constant) would be handy to know. It would be even more convenient if the partial could be adjusted to show the amount of node voltage change caused by a one percent variation of element value. This adjusted partial is called a sensitivity coefficient. Sensitivity coefficients allow a designer to pinpoint the elements which must have highly precise values and the ones which can have wide tolerances. ECAP DC ANALYSIS will produce the partial derivatives and sensitivity coefficients for all node voltages with respect to all resistances, all fixed-magnitude voltage and current sources, and the transconductances of all dependent sources. The partials and sensitivities are exact solutions and not approximations.

To produce a DC ANALYSIS partial derivative and sensitivity coefficient output for a problem, the card

```
/1      7
        SENSITIVITIES
```

must be placed in the problem data deck somewhere between the DC ANALYSIS card (or MODIFY card for modified solutions) and the EXECUTE card. Also, the output specification code SE must appear on the PRINT card. ECAP will <u>not</u> vary an element value over a range with the p1 (p2) p3 method discussed earlier, and at the same time produce partial derivatives and sensitivities. The following example shows how to specify sensitivities output, but the data cards have been deleted.

```
/1          7
            DC ANALYSIS
 B xx       Nxxxxxxxxxxxx
 T xx       Bxxxxxxxxxxxx
            SENSITIVITIES
            PRINT, NV , MI , SE
            EXECUTE
 C
            MODIFY
 B xx       R = xxxxxxxxxx
 T xx       GM = xxxxxxxxx
            SENSITIVITIES
            EXECUTE
            END
 /*
```

Note in the preceding example that the PRINT card carried over into the modified solution, but that a new SENSITIVITIES card was needed to carry over the SE entry on the PRINT card.

After the tolerance limits of all elements in a circuit are specified, the resulting circuit could have node voltages that differ from those calculated with exact element values. For a given node, the worst case voltages would be the largest

and smallest voltages that could occur. This solution is obtained by observing the sign on each partial, substituting appropriately either the largest or smallest allowable element values in the circuit equations, and then solving the equations for the node voltage. ECAP will produce both the worst case maximum and minimum voltages for all circuit nodes (except the reference node); but only if the tolerances of the elements are specified in the 'B' and 'T' cards. Since the sign of the partial determines whether the maximum or minimum value of an element is substituted into the equations, ECAP recalculates the partials for both the worst case maximum and minimum solutions with all elements at their maximum or minimum value as appropriate. Since the partial derivatives are not constants; if the sign of a 'new' partial is opposite to the sign of the corresponding 'old' partial, the worst case solution is not a true worst case. In this case, the program will print an error message that indicates that the worst case solution is not a true worst case. If the partials do not have a sign change, the worst case solution is a true worst case.

The 'B' and 'T' cards must be modified to show element tolerances. The program user has two options in specifying tolerances. The first option has data groups of the form:

p1 (p2,p3)

where

p1 is the nominal value of the element,

p2 is the minimum value of the element,

p3 is the maximum value of the element,

An example of this option in element specification would be BETA = 14 (12 , 17.3) . The second option has data groups of the form:

p1 (p4)

where:

p1 is the nominal value of the element,

p4 is the decimal value of the percent tolerance.

An example of this option in element specification would be E = - 14 (.10). The tolerance on E is 10% in this example. Some examples that show both options are:

```
/1      7
 B1     N(0,1), G = .011 (.05), E = 8, I = 5(2.5,7)
 B2     N(1,0), R = 3.33 ( 2, 3.5 ) , I = 4
 T1     B(1,2) , BETA = 2 ( .20 )
```

The same options in specifying tolerance can be used on the abbreviated data cards of a MODIFY solution. From the preceding examples it is seen that not all elements must have tolerances specified, but if any tolerances are omitted in the data the solutions derived from the tolerances will be valid only when those tolerance-elements unspecified have their exact value.

In addition to the tolerance information on the 'B' and 'T' cards, the user must include the solution control card

```
/1    7
      WORST CASE
```

somewhere in the problem deck (before the EXECUTE card). The PRINT card must also specify the solution output code WO.

If worst case solutions are needed for only a few nodes, the WORST CASE card can be of the form:

```
/1     7
       WORST CASE, X , Y, Z , Q
```

where X,Y,Z, and Q are the numbers of the nodes of interest.

In the construction of circuits, the individual elements are randomly selected from batches of elements. The statistical properties of each batch is specified by its nominal value (mean) and standard deviation. If all batches are known (or assumed) to be normally distributed, then ECAP can estimate the standard deviation of the node voltages of a circuit constructed from the batches of elements. If the 'B' and 'T' cards specify tolerance by the decimal value of the percent, the decimal percent represents three times the standard deviation of the batch. Thus if R = 150 (.03) , then the standard deviation of the batch would be (150 x .03)/3 = 1.5 ohm. If the 'B' and 'T' card specify tolerance by maximum, nominal, and minimum value; the difference between the maximum and nominal value is three times the standard deviation of the batch. Because of the restriction to normal probability distributions, the difference between the maximum and nominal values must equal the difference between the nominal and minimum values. In other words, the maximum and minimum must be symmetric with respect to the nominal value. ECAP will estimate standard deviations of node voltages if only a few element tolerances are specified, but will not if an element value variation is attempted at the same time.

To produce a standard deviation output from ECAP; in addition to symmetric data tolerances, the solution control card:

```
/1      7
        STANDARD DEVIATION
```

must be included somewhere in the problem deck and the solution control code ST must appear on the PRINT card. If either a standard deviation or worst case solution is wanted in a MODIFY solution; the solution control cards WORST CASE and STANDARD DEVIATION respectively must be included with the modified data.

Figure 3-7 shows the problem of Figure 3-6 with tolerances specified and several different solutions requested. Note that one MODIFY solution includes an element value variation; but no request for X sensitivities, worst cases, or standard deviations. Such a request would be invalid. The variation solution uses the nominal values of the elements as exact values. Notice also that the worst case solution will include worst cases for only nodes one and four. The first modification will produce only partial and sensitivity coefficients; since the solution control cards WORST CASE and STANDARD DEVIATION are not included.

```
/1      7
 //EXAMPLE  JOB (06W12345670001,2,2),URSTUCK,MSGLEVEL=1
 //JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001 DD SYSOUT=A,DCB=RECFM=UA
 //FT05F001 DD *
 C
        DC ANALYSIS
 C
 B1     N(0,1), R= 5(.10), E= .15 E +02 (.05)
 B2     N(1,2), R = 8 (7,9)
 B3     N(1,4), G = .01 (.05)
 B4     N(0,2), R = 100 (.05)
 B5     N(0,3), R = 86 (80,92)
 B6     N(3,4), R = 4 (3,5)
 B7     N(0,4), R = .51 E +02 .50 E +02, .52 E +02)
       *         T = - .2 E +01 ( .07)
 T1     B(5,4), GM = -.43 (.06)
 T2     B(4,5), BETA = 05(-6,-4)
        WORST CASE , 1 , 4
        SENSITIVITIES
        STANDARD DEVIATION
        PRINT, NV , SE , WO , ST
        EXECUTE
 C
        MODIFY
 B1     R = 2 (.05)
 T1     GM = .8 (.7,.9)
        SENSITIVITIES
        EXECUTE
 C
        MODIFY
 B2     R = 15
 T2     BETA =-10 (20) +10
        EXECUTE
        END
 /*
```

Figure 3-7. Example Problem of Figure 3-6 with Additional Solution Outputs Requested.

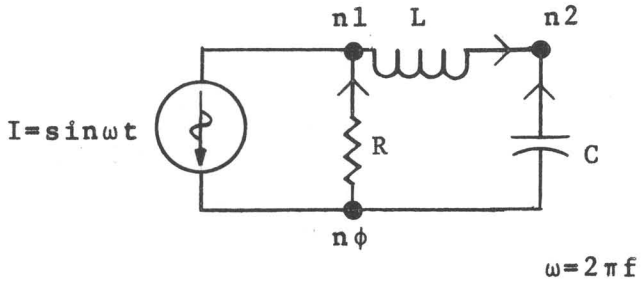
CHAPTER 4

INSTRUCTION GUIDE FOR SOLVING AC CIRCUITS WITH ECAP/360

Basic Considerations

ECAP/360 discribes an AC circuit with a set of equations derived from Kirchoff's nodal law: 'the sum of currents into a node must be zero'. The set of nodal equations is converted into a matrix equation of the form [Y][E] = [I].[Y] is called the nodal admittance matrix and [I] is called the equivalent current vector. Figure 4-1 shows a simple circuit and its associated equations. The term f is frequency in hertz. The solution of the set of nodal equations will yield the node voltage matrix, [E]. The currents in the elements can be found easily once the node voltages are known by applying Ohm's law for AC circuits. The writing of the nodal equations and the application of Ohm's law tacitly assumes that a positive direction of current flow and the node connections have been specified, along with positive conventions for voltage and current sources. While an individual might 'cut some corners' in specifying these conventions if he were writing the circuit equations himself; ECAP requires a complete and accurate description of assigned conventions. To simplify the problem of specifying positive conventions, a 'Standard Circuit Branch' has been divised. Figure 4-2 shows the AC Analysis 'Standard

Circuit



Differential Nodal Equations

$$\frac{e_1}{R} + C\frac{de_1}{dt} - C\frac{de_2}{dt} = \sin\omega t$$

$$-C\frac{de_2}{dt} + C\frac{de_2}{dt} + \frac{1}{L}\int_0^t e_2 dt = 0$$

Laplace Nodal Equations

$$(1/R + Cs)e_1 - Cs\ e_2 = \frac{\omega}{s^2+\omega^2}$$

$$-Cs\ e_1 + (Cs + \frac{1}{Ls})e_2 = 0$$

Matrix Nodal Equation

$$\begin{bmatrix} (1/R+j\omega C) & -j\omega C \\ -j\omega C & j(\omega C-1/\omega L) \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} -j1 \\ 0 \end{bmatrix}$$

Figure 4-1. A Simple AC Circuit and its Equations

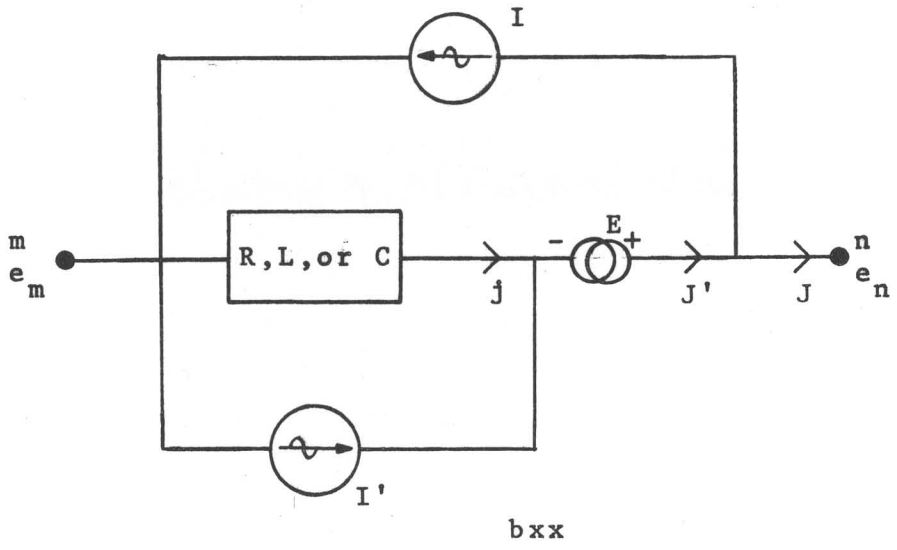


Figure 4-2. ECAP AC ANALYSIS 'Standard Circuit Branch'

Circuit Branch' with its associated conventions. The definitions for the items shown in Figure 4-2 are:

R,L,or C	is the branch element resistance, inductance, or capacitance in ohms, henrys, or farads respectively,
E	is the branch fixed-magnitude voltage source,
I	is the branch fixed-magnitude current source,
bxx	indicates branch number xx,
j	is the current flow in the branch element,
I'	is the branch dependent current source (which is controlled by the voltage across the element of another branch)
J'	is defined as the element current,
J	is defined as the branch current,
m	is the 'from' node number of the branch,
n	is the 'to' node number of the branch,
e_m	is the node voltage at node m,
e_n	is the node voltage at node n,
f	is the circuit frequency in hertz

All positive conventions are shown in Fig. 4-2 with arrows or polarity (+ & -) marks that indicate the in-phase (zero degrees) direction of current and voltage. It should be noted that the program user can think of either sine wave or cosine wave excitation as in-phase for the circuit; but once sine wave is chosen as reference, it applies to <u>all</u> the branches of the circuit. The phase angle is also determined with respect

to a user specified circuit reference. For example; if the user specifies a current source to be at an angle of thirty degrees, he has specified the reference angle for that source. In Fig. 4-1; cosine was the reference, as can be seen from the matrix equation. Positive element and branch current flow from the 'from' node of the branch. Once the positive direction of current in a branch is specified by the program user; all other conventions for that branch are implied by the 'Standard Circuit Branch'. Every branch must have a non-zero resistance, capacitance, or inductance. A branch cannot have combinations of elements (like a resistor and a capacitor); each element must be assigned as a separate branch. Since each branch must have a non-zero element; every source associated with the branch has a finite impedance, but this impedance can be very large or small. No sources can be specified independent from the branches of the circuit. If a source is not wanted in a specific branch, no data for that source is provided to ECAP. ECAP will assume a source current or voltage to be zero in the absence of instructions. If source data is included for a branch, the user can assign both the rms magnitude and phase angle. Those persons who are already familiar with ECAP/360 DC ANALYSIS will recognize that the 'Standard Circuit Branch' in AC ANALYSIS is almost the same as it was in DC ANALYSIS.

Every node and branch of the circuit must by uniquely numbered. One node must be assigned as the zero-voltage (reference) node. The reference node is assigned the number zero.

All node numbers from zero to the highest node number must be assigned. If there are fifteen nodes in a circuit; the highest node number is fourteen, and all numbers between zero and fourteen must be assigned. No node or group of nodes can be isolated (i.e.; without some connection to the reference node). Branch numbers must run from one to the highest branch number. Branch and node numbers are arbitrarily assigned by the program user, as are in-phase current directions. Figure 4-3 shows a circuit with all the convention assignments that would be needed to describe the circuit for ECAP. Notice that the 'Standard Circuit Branch' need not be completely shown for each branch, only the items physically present in the network need be shown.

All the labeling and assigning of conventions to a circuit may seem to be a big nuisance, but it is not nearly as bad as trying to solve a circuit with many nodes and branches by hand. ECAP/360 will handle circuits with up to:

50 nodes, not including the reference node,
200 branches,
200 fixed-magnitude and phase voltage sources,
200 fixed-magnitude and phase current sources,
200 dependent current sources,
NO dependent voltage sources,
25 mutual inductances.

If a circuit has one or more mutual inductance specified, the total number of inductances in the circuit (not including the mutual inductances) cannot exceed fifty. Put another way; the

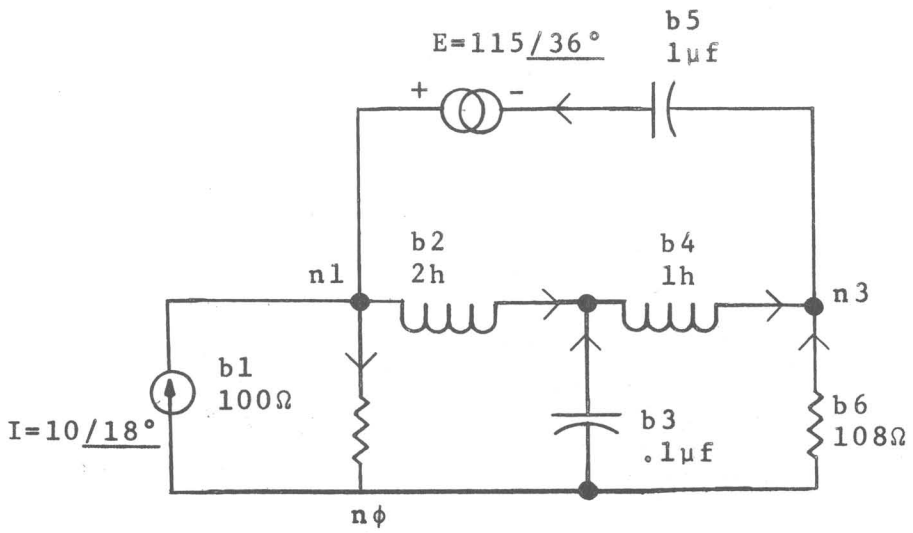


Figure 4-3. Example AC Circuit with Branch Conventions Assigned

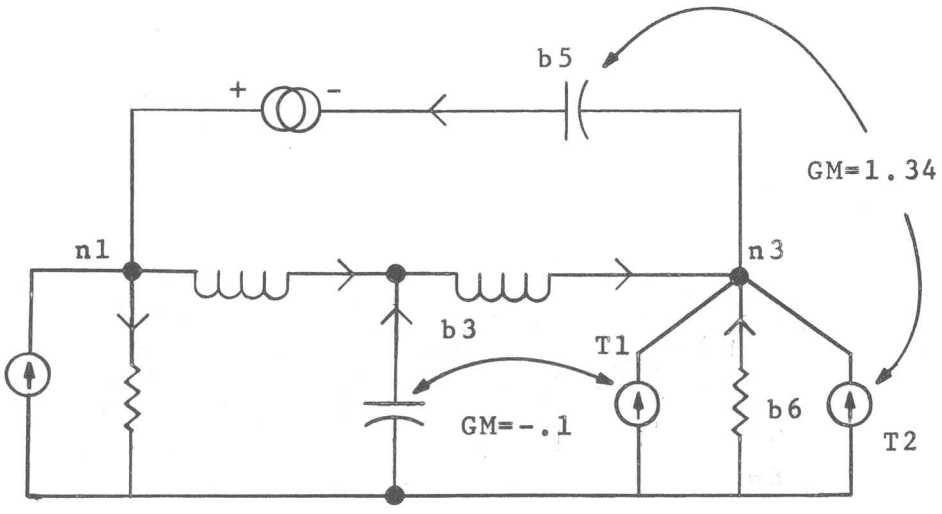


Figure 4-4. Example Circuit of Figure 4-3 with Two Dependent Sources Added

circuit cannot have more than twenty five transformers with two windings on each transformer.

The dependent current source specified in ECAP is really an isolating amplifier. Even though a dependent voltage source cannot be specified; if one is needed, it can be converted into a Norton equivalent dependent current source by the program user. A dependent current source is physically located in its 'to' (controlled) branch, but its current flow is regulated by the voltage across the resistor, inductor, or capacitor in another branch. The regulating (controlling) branch is the 'from' branch of the dependent source. The link between the 'from' and 'to' branch can be specified by a transconductance GM. GM must be a real number. For example; if the voltage across the 'from' branch is 5 volts rms at an angle of 18 degrees (5/18), the voltage source in the 'from' branch 1/18, and the transconductance between the 'from' and 'to' branches is 3.1 mhos; then the current in the dependent current source in the 'to' branch is: (3.1)(1/18 + 5/18) = 18.6/18 amperes rms. If and only if the 'from' branch contains a resistor can the link between the 'from' and 'to' branches of a dependent current source be specified as a current gain, BETA. BETA must be a real number when it is specified. For a resistive 'from' branch, GM and BETA are related by the formula: GM = BETA/R_f where R_f is the resistance of the 'from' branch. Figure 4-4 shows the circuit of Fig. 4-3 with the addition of two dependent current sources. Some of the branch labels in Figure 4-4 have been omitted for clarity. Notice that a branch may have more than one dependent source.

It is <u>not</u> possible for a branch to have more than one fixed-magnitude and phase voltage (or current) source. Also note that if the 'from' branch has a fixed-magnitude and phase voltage source, the controlling voltage for the dependent source is the voltage across the element and not the voltage across the branch.

Mutual inductances couple two windings of a transformer together magnetically. The coefficient of coupling (k) is defined by the formula:

$$k = \frac{|L_m|}{\sqrt{L_p L_s}}$$

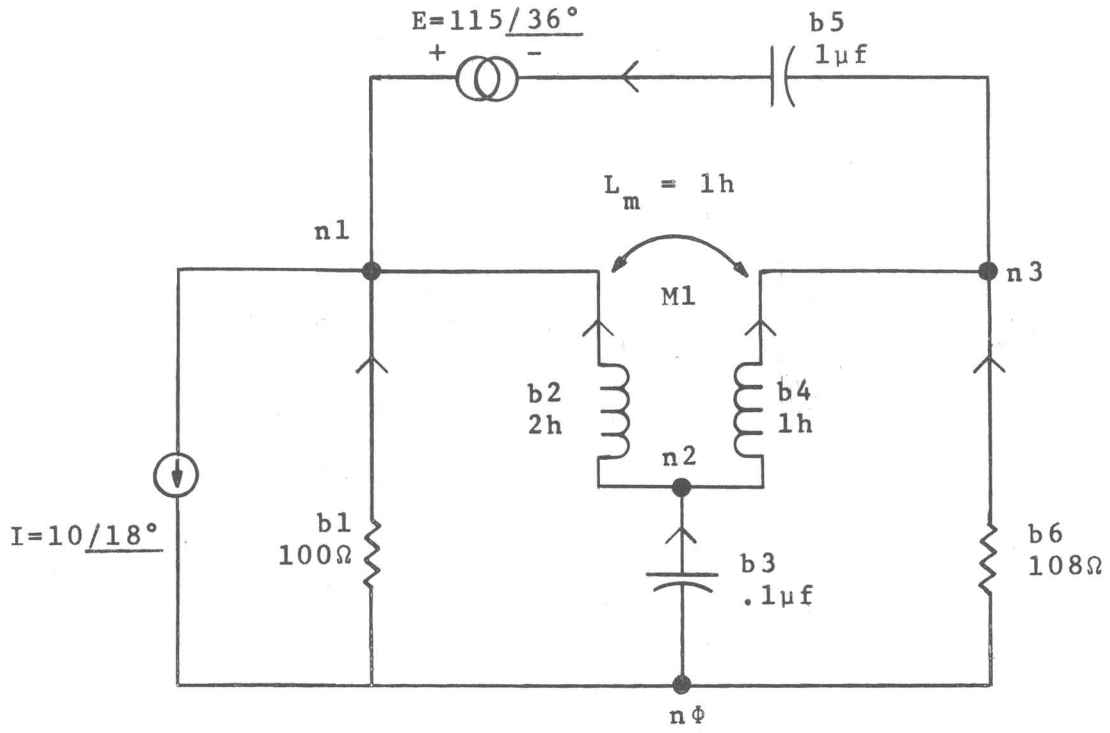
where:

L_m is the mutual inductance in henrys,

L_p is the inductance of the primary winding in henrys,

L_s is the inductance of the secondary winding in henrys.

The coefficient of coupling for a transformer must be greater than zero, but less than one. When L_m is positive, an increase of current into the 'from' node of the primary causes a positive voltage at the 'to' node of the secondary. If L_m is negative, the positive voltage would be at the 'from' node of the secondary. Figure 4-5 shows an example circuit with mutual inductances. Note that the primary and secondary windings are assigned as independent branches, but the mutual inductance is not a branch.



frequency = 60 hertz

$k = |1|/\sqrt{2} = .707$

Figure 4-5. Example Circuit with Mutual Inductance Coupling

If the coefficient of coupling were allowed to equal one, the transformer would be called 'perfect'. ECAP will not solve circuits with 'perfect' transformers.

Table 4-1 shows the definition of several parameters. It should be remembered that all parameters except GM are complex (real and imaginary parts). Table 4-1 assumes that branch z has 'from' node x, 'to' node y; and the dependent source in branch z has 'from' branch q and transconductance GM_{qz}. Z_z is the impedance of branch z. In Table 4-1, it is important to note the definition of branch current, element current, and branch power loss. Specifically, it is the method of calculating branch power loss that justifies the Figure 4-2 representation of the 'Standard Circuit Branch'. The 'Standard Circuit Branch' of Figure 4-2 (which applies to ECAP/360) is not the same as the one shown in the IBM ECAP/1620 USER'S MANUAL; though this manual in general applies to ECAP/360 also.

Basic Instructions Required to use ECAP/360 AC Analysis

ECAP/360 at the present is available from disc storage at the KSU Computing Center. Instructions to the computer can be broken into eight classifications:

1. Commands to get the program from storage (5 cards),
2. User circuit data,
3. Solution Controls,
4. Output Specifications,
5. Commands,

Parameter	Symbol	Defining Equation	Available as an ECAP/360 Output	Output 'PRINT' Code
Node x voltage	e_x	Solution of System Eq.	YES	NV
Branch z Voltage Source	E_z	User Data	NO	
Branch z Voltage	V_z	$V_z=e_x-e_y$	YES	BV
Element Voltage	V'_z	$V'_z=V_z+E_z$	YES	CV
Branch z Current Source	I_z	User Data	NO	
Branch z Dependent Source	I'_z	$I'_z=GM_{qz}V_q$	NO	
Branch z Impedance	Z_z	Computed from User Data	NO	
Impedance Current	j_z	$j_z=V'_z/Z_z$	NO	
Element Current	J'_z	$J'_z=j_z+I'_z$	YES	CA
Branch z Power Loss	P_z	$P_z=J'_z\ V'_z$	YES	BP
Branch z Current	J_z	$J_z=J'_z-I_z$	YES	BA

Table 4-1. ECAP AC ANALYSIS definitions for Parameters.

6. System Controls,

7. Comments,

8. Termination of job instructions (2 cards).

All input to the computer is by punched cards, and all output from the computer is printed. There is no punched card output option. In the remainder of these instructions; the standard eighty entry data card will be represented by a /1 above the position where column one would be, and a 7 above column seven. Imbedded blanks are assumed to be one character entry long unless a column number entry is shown above the character following the blanks. No entrys are allowed in columns 73 - 80. A sample of two comment cards (one after the other) by this system fo representation would be:

```
/1    7
 C    A.C.SPECIAL TEST PROGRAM
 C    THREE-PHASE TRANSMISSION NETWORK
```

There are five special cards required to get the program from the disc. They are <u>not</u> part of ECAP, but are control cards required by the Computing Center for orderly operation. All five of them <u>must</u> be punched in the EBCDIC (extended binary code) character set. The EBCDIC characters are the ones punched by an IBM 029 Card Punch, or multiple punched on an IBM 026 Printing Card Punch. The 026 Printing Card Punch normally punches BCD (binary code) characters. The first card must be a 'job card' exactly the same as for other programs submitted to the Computing Center. At present, this card is completely specified in KSU Computing Center Notice #52, dated 27 February 1968. An

abbreviated form of this card which 'will work' is:

```
                1   1
      /1        2   6
       //jobname  JOB (pano,time,lines),yername,MSGLEVEL=1
```

where:

// must appear in columns 1 and 2.

jobname is 1 - 8 characters which name the job, the first of which must be alphabetic,

JOB must start in column 12,

(must appear in column 16,

pano is your job charge number,

time is the maximum running time in integer minutes,

lines is the maximum number of lines of printed output in integer thousands of lines,

yername is your own name with no imbedded blanks.

A sample of a job card would be (with a false job number):

```
                1    1
      /1        2    6
       //ALLERROR  JOB (06W190978Q001,5,4),URSTUCK,MSGLEVEL=1
```

If the running time of the problem exceeds the 'time' entry on the job card; the computer will automatically terminate the job. The job will also be terminated if the number of 'lines' is exceeded. Either the 'time' or 'lines' entry (or both) can be omitted. If 'time' is omitted, the data group inside the parentheses becomes (pano,,lines). There can be no blanks between the commas. The computer will assume a 'time' entry of 1 minute. If the 'lines' entry is omitted, the data group inside the parentheses becomes (pano,time). The computer will assume a 'lines' entry of 1. If both 'time' and 'lines' are omitted; the entry becomes (pano), and the assumed entrys are the same.

The remainder of the five special command cards as presently specified are:

```
/1      7
 //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
 //FT05F001 DD *
```

These five special cards must be the first five cards in any ECAP problem deck, and they must be in the order presented.

Two special cards are used to terminate an ECAP job, and they are the last two cards in an ECAP problem deck. In proper order, they are:

```
/1      7
        END
 /*
```

Comment cards can be placed anywhere after the first five cards and before the last two cards of the problem deck. Column 1 of a comment card must have the character C punched in it. Any remark can be included in columns 2 - 72. Two examples of comment cards were included as an earlier example.

The remaining cards to be discussed will be assumed to be punched in EBCDIC, but a method for using BCD punched cards will be shown later.

Four types of data cards are used to describe the problem circuit to ECAP. The first type specifies the frequency of the circuit in hertz. An example of this type of card would be:

```
/1      7
        FREQUENCY = 60
```

If the frequency is not specified, ECAP will reject the problem.

The second type of data card specifies all the data for one branch, except the dependent source data. This card is called a 'branch' or just 'B' card. The standard forms for the 'B' cards are: for resistive elements,

```
 /1      7
  Bxx    N(ww,zz),R=rr,E=ee/aa,I=ii/bb
```

for inductive elements,

```
 /1      7
  Bxx    N(ww,zz),L=ll,E=ee/aa,I=ii/bb
```

for capacitive elements,

```
 /1      7
  Bxx    N(ww,zz),C=cc,E=ee/aa,I=ii/bb
```

where

B indicates that it is a 'B' card,

xx is the number of the branch, and can appear in any of the columns 2 - 5,

N indicates the node connection of the branch,

ww is the 'from' node of the branch,

zz is the 'to' node of the branch,

rr is the resistance of the branch in ohms (if it is a resistive branch),

ll is the inductance of the branch in henrys (if it is an inductive branch),

cc is the capacitance of the branch in farads (if it is a capacitive branch),

ee is the rms magnitude in volts of the fixed-magnitude and phase voltage source of the branch,

aa is the phase angle in degrees of the fixed-magnitude and phase voltage source,

ii is the rms magnitude in amperes of the fixed-magnitude and phase current source of the branch,

bb is the phase angle in degrees of the fixed-magnitude and phase current source.

If either aa or bb are zero, they can be omitted from the 'B' card. If ee is zero, the complete entry E=ee/aa can be omitted; as can I=ii/bb if ii is zero. The entrys rr,cc, and ll cannot be zero; but they can be very small. If the conductance of a resistive branch is known; the entry R=rr can be replaced with G=gg, where gg is the conductance of the resistor in mhos.

The third type of data card specifies dependent current source data. It is called a 'T' card after its column one entry. The standard form of a 'T' card is

```
/1      7
 Txx    B(bb,dd),BETA=tt
```

or

```
/1      7
 Txx    B(bb,dd),GM=kk
```

where

T indicates a dependent source, 'T' , card,

xx is the number of the dependent source,

bb is the 'from' branch of the dependent source,

dd is the 'to' branch of the dependent source,

tt is the current gain of the dependent source (only if the 'from' branch is resistive),

kk is the transconductance of the dependent source.

Only one 'T' card is needed for each dependent source. In general, the 'from' branch element can be an inductor, capacitor, or resistor. The dependent source can be specified with a BETA only if the 'from' branch is resistive; but it can be specified with a GM for all cases. In the case of a resistive 'from' branch, GM and BETA are related by the formula $GM = BETA/R_c$ where R_c is the resistance of the 'from' branch. ECAP restricts GM (and BETA when used) to real numbers. The dependent sources are uniquely numbered in order from one to the highest number. No number between one and the highest can be missing.

The fourth type of data card specifies mutual inductances and is called an 'M' card. The standard form of an 'M' card is:

```
/1      7
 Mxx    B(bb,dd),L=mm
```

where

M indicates a mutual inductance, 'M', card,

xx is the number of the mutual inductance,

bb is the branch number of one of the inductors coupled by the mutual inductance,

dd is the branch number of the other inductor coupled by the mutual inductance,

mm is the inductance of the mutual coupling in henrys.

The branch numbers bb and dd can be interchanged with no resulting difference. The mutual inductances are uniquely numbered in order from one to the highest number. No number

between one and the highest can be missing. ECAP will handle up to twenty five mutual inductances with the restriction mentioned earlier.

The node and branch numbers that appear on the data cards must be non-negative integers. All other data entrys can be up to eight digits, with or without a decimal, and with or without an E followed by an integer that indicates an exponential power of ten. The largest number accepted by ECAP is 7 E +75, which is 7 followed by 75 zeros. The exponent can also be a negative integer. Sometimes the program output uses D instead of E, but it means the same thing. Using D on an input data card to indicate an exponent is not allowed.

Several examples of 'B', 'M', and 'T' cards can be developed from the preceding discussion. Note that not all entrys are necessary on a 'B' card if either the E or I source are missing in the branch. Also, the angle for E on card B13 is missing, so ECAP will assume it is zero degrees.

```
/1      7
 B7     N(0,3),L=.10, E= 115/45, I= 3/-26
 B13    N(4,5),L= 1 E -01, E= 1E+2
 M 1    B(7,13), L = .05
 B 3    N(1,6), R = 100
 B 4    N(2,6), R = 57 , I =-15/199
 T 1    B(7,22), GM = -.22345678
 T  2   B(3,16), BETA = .113
 C      NOTICE THAT T 2 HAS FROM BRANCH B 3
 C      WHICH IS RESISTIVE, SO BETA CAN BE
 C      USED.
```

It can be seen from the preceding examples that the entrys can have a lot of imbedded blanks (an imbedded blank is a blank entry between two other characters) and be rather free in order.

The entrys in column one must be 'M', 'B', or 'T' for data cards (other than a FREQUENCY card). Entering the N or B in column seven is 'good form', but it can occur later than column seven (but not before).

Sometimes the columns 7 - 72 may not provide enough space to completely specify all the information on a 'B' card. The data for that card can be continued on a second (or even third) card by placing an * in column six, and punching the remaining data in columns 7 - 72 of the continuation card. The continuation card (cards) for a 'B' card must immediately follow that 'B' card in the problem deck. Since no entry is included in columns 1 - 5 to indicate which branch is being continued, the continuation card should be marked with a pen or pencil to show the branch number. An example of a 'B' card and two continuations would be:

```
/1    67
 B 1   N(3,4), C = 10.0 E -6,
       *       I = .23456789 E -01 / 135 ,
       *       E = .98765432 /122.7
 C  NOTICE THAT THE ANGLE ON E HAS A DECIMAL
```

'B' cards can be continued. 'T' and 'M' cards cannot be continued.

Solution Output Specification cards have the word PRINT starting in column seven followed by letter codes that control solution outputs. Only one PRINT card is needed for each problem. The letter code for output specification is shown in Table 4-2. Some typical PRINT cards would be:

```
/1      7
        PRINT, NV , BA
        PRINT,NV,BV,BP
        PRINT,NV
        PRINT,NV,BV,CV,CA,BA,BP
```

Voltage and current solutions are represented in polar form by ECAP.

Code	ECAP Solution Output
NV	NODE VOLTAGES
BV	BRANCH VOLTAGES
CV	ELEMENT VOLTAGES
CA	ELEMENT CURRENTS
BA	BRANCH CURRENTS
BP	BRANCH POWER LOSSES

Table 4-2. Solution Output Codes.

Two command cards not previously mentioned are needed before a complete problem input deck can be described. The first command must appear in the deck just ahead of the data cards and specifys to ECAP that the problem to be solved is an AC circuit. This card is:

```
/1      7
        AC ANALYSIS
```

The second command card needed is the last card in the problem deck (but still before the END and /* cards). It commands the program to

```
/1      7
        EXECUTE
```

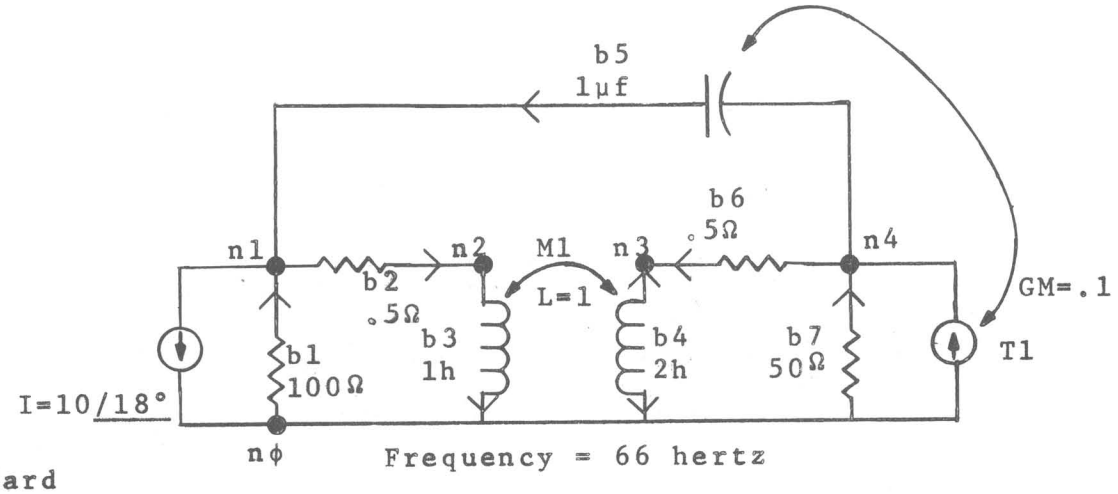
the solution.

Before considering an example of a complete problem deck, it should be mentioned that ECAP problems can be 'stacked' in batches on one job card. Instead of placing the END and /* cards after the EXECUTE card of the first problem, another AC ANALYSIS card followed by its data and EXECUTE card is included instead. Several problems can be stacked, and the END and /* cards are placed after the EXECUTE card of the last problem. All the problems are solved separately and the outputs are identified by also reproducing the input data along with the solutions. Some time is saved by not getting the ECAP program from disc storage for each solution. The number of problems that can be stacked and solved on one job card is limited only by the 'time' and 'lines' entrys on the job card. Figure 4-6 shows an example of converting a circuit to a problem input deck. The card classifications and card formats are reproduced in Table 4-3. Figure 4-6 includes the card classification number of each card for quick reference to Table 4-3.

ECAP has a built in check on the accuracy of its AC solutions. Since the current at each node should sum to zero, the program computes those sums. The currents are known to the program by their real and imaginary parts; so the reals are summed separately, and the imaginarys are summed separately. The sums will not always be exactly zero since ECAP is limited to sixteen significant figure accuracy (eight in input and output) in its calculations. After computing the difference from zero at each node for the reals, ECAP sums the absolute value of these differences at all nodes. This sum is called the real current

Card Class. Number	Card Class	Card Name	Card Format /1 7
1	Commands to get the Program	jobcard	(as currently specified)
		None	//JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
		None	//STEP1 EXEC PGM=ECAP
		None	//FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
		None	//FT05F001 DD *
2	User's Circuit Data	'B'	Bxx N(y,z), R=r
		'B'	Bxx N(y,z), L=l
		'B'	Bxx N(y,z), C=c
		'B'	Bxx N(y,z), L=l , E=e/d , I=i/f
		'B'	Bxx N(y,z), R=r , I=i
		Continuation	* Remaining data of a 'B' card
		'T'	Tqq B(a,b), GM = gg
		'T'	Tqq B(a,b), BETA = mm
			Note: BETA can be used only if 'from' branch is resistive
		'M'	Mtt B(j,k), L = p
			Note: Coefficient of Coupling can not equal one.
3	Solution Control	'Freq'	FREQUENCY = ff
4	Output Spec.	'Print'	PRINT,NV
		'Print'	PRINT,NV,BV,CV,CA,BA,BP
5	Commands	None	DC ANALYSIS
		Execute	EXECUTE
6	System Control	None	*BCD
		None	*EBCDIC
7	Comments	Comment	C any comment necessary to
		Comment	C understand the problem
8	End of Job Instruc.	End	END
		None	/*

Table 4-3. ECAP/360 AC ANALYSIS Input Cards




```
Card
Classification
Number                                  Program Cards

          /1      7
  1        //EXAMPLE   JOB (06W1234567Q001.2.3),JJJONES,MSGLEVEL=1
  1        //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
  1        //STEP1 EXEC PGM=ECAP
  1        //FT06F001 DD SYSOUT=A,DCB=RECFM=UA
  1        //FT05F001 DD *
  7        C
  7        C       EXAMPLE AC ANALYSIS PROGRAM
  5                AC ANALYSIS
  2        B1      N(0,1), R = 100 , I = 10 / 18
  2        B2      N(1,2), R = .5
  2        B3      N(2,0), L = 1.0
  2        B4      N(0,3), L = 2.0
  2        M1      B(3,4), L = 1.0
  7        C       NOTE THAT THE COEFFICIENT OF COUPLING
  7        C       IS .707
  2        B5      N(4,1), C = 1.0 E -06
  2        B6      N(4,3), R = .5
  2        B7      N(0,4), R = .50 E +02
  2        T1      B(5,7),GM = .1
  3                FREQUENCY = 66
  4                PRINT,NV,CA,BP
  5                EXECUTE
  8                END
  8        /*
```

Total Execution Time for this problem was: 0.38 minutes.
Total Amount of Printed Output was: 8 pages, including the pages due to the HASP Log System of the KSU Computing Center.

Figure 4-6. Example Circuit and Problem Deck for ECAP/360 AC ANALYSIS.

unbalance of the solution. Though the current unbalance is never printed out, if it is larger than .001 ampere an error message will be printed along with the solution obtained. The summing of unbalance currents is repeated for the imaginarys if the reals unbalance is less than .001. The comparison with .001 ampere is repeated for the imaginary current unbalance with an error message resulting if .001 ampere is exceeded. This error usually occurs when the impedances in the circuit are spread over several orders of magnitude.

Sometimes it is inconvenient to punch cards with EBCDIC characters. If a group of cards, other than the first five cards in the deck, have been punched with BCD characters; they can be used if the card:

```
/1      7
 *BCD
```

is inserted in the deck immediately ahead of the BCD punched group of cards. If later in the deck EBCDIC punched cards are included (say in another problem), the card:

```
/1      7
 *EBCDIC
```

converts the reading interpretation back to EBCDIC characters. In the absence of instructions, ECAP assumes the cards are all EBCDIC. If this assumption is wrong, a battery of error messages will be produced. The error messages will indicate by card number (in the order it appears in the deck) the card (cards) in error and also pinpoint the column of that card where an incorrect character occurs.

It was mentioned earlier in this chapter that an ECAP program also exists for the IBM 1620. Since ECAP/360 is a greatly expanded version of ECAP/1620, persons specifically interested in using ECAP/1620 should become familiar with both the ECAP/1620 USER'S MANUAL and the ECAP/1620 SYSTEM MANUAL. An object deck for ECAP/1620, along with all the ECAP/1620 manuals, are available at the operations office of the KSU Industrial Engineering Department's IBM 1620.

Additional ECAP AC ANALYSIS Solution Features

It was mentioned earlier that ECAP solves AC problems by forming the matrix equation [Y][E] = [I] and then solving the matrix equation. If the output specification code MI is included in the PRINT card, the program will print out the matrices [Y] and [I]. An example output due to including MI on a PRINT card would be:

	ROW NODE	COL NODE	NODAL ADMITTANCE MATRIX	
REAL	1	1-2	.23456789E-01	.00000000E-99
IMAG			.12345678E-02	.98765432E-01
REAL	2	1-2	.67891234E+01	.98765432E+02
IMAG			.54321987E+02	.89123456E+03

	NODES	EQUIVALENT CURRENT VECTOR	
REAL	1-2	.20000000E-02	.00000000E-99
IMAG		.00000000E-99	.00000000E-99

Row one of the admittance matrix and the current vector correspond to the terms in a nodal equation written at node one.

Likewise, row n would be an equation written at node n.

Sometimes a user may want to repeat a solution with only a few values changed. It would be inconvenient to punch a whole new deck of cards for the 'new' problem. A short method consists of placing the card

```
/1      7
        MODIFY
```

after the EXECUTE card of the 'old' problem; and then new 'B', 'T', or 'M' cards in an abbreviated form for those branches and dependent sources that are to be changed. Only the value of elements can be changed. Node connections or the number of nodes cannot be changed with the MODIFY procedure. If the resistance of branch 2 was to be changed to 4 ohms from 8 ohms, the card:

```
/1      7
 B2     R = 4
```

would be included after the MODIFY card. Other examples of abbreviated data cards would be:

```
/1      7
 B3     I = 10 / 35
 B7     R = 11.1 , E = 18 / 180
 B9     E = 26 / - 45
 T1     GM = 1.00
 M2     L = .33
        FREQUENCY = 100
```

It is seen that a data card in its abbreviated form must identify the branch and include the parameter to be changed. The nodal connections from the 'old' problem is carried over to the 'new' problem. If the voltage or current source are changed; both the rms magnitude and the phase angle must be restated. If the frequency is to remain the same, a new FREQUENCY card would <u>not</u>

be needed. No new PRINT card is needed, as the 'old' PRINT card carrys over to the MODIFY solution. The MODIFY solution is a completely new solution and not an approximation from the 'old' solution; therefore the problem can be modified several times without a deterioration in accuracy. Each successive modification must have its own MODIFY, data, and EXECUTE cards. The END and /* cards are placed after the EXECUTE of the last MODIFY solution to terminate the job. If the 'from' branch of a dependent source is resistive and the control specified on the associated 'T' card is by a BETA; then if the resistance of the 'from' branch is changed in a MODIFY solution, the BETA must be restated. ECAP converts the BETA to a GM, and this calculated GM is carried over to the MODIFY solution unless restated. In a later section on variation of parameter values, this restriction should be remembered as no restating of BETA is then possible. Not more than 50 parameters (R,C,L,E,I, & GM) total can be changed at the same time.

By using the MODIFY method, the frequency response of a network could be found by having ECAP solve the circuit several times with only the frequency changed. However, ECAP/360 offers two 'shortcut' methods to find the frequency response. The first method uses a data group on the FREQUENCY card which is of the form:

p1 (+p2) p3

where

p1 is the lowest frequency desired,

p3 is the highest frequency desired,

p2 is the integer number of uniformly incremented steps of frequency desired from the lowest to the highest frequency.

If the entry were FREQUENCY = 10 (+9) 100 , complete solutions (not approximations) would be calculated at 10, 20, 30, ... 90, 100 cycles per second. The second frequency response method lets the user increase frequency in logarithmic increments. The data group on the FREQUENCY card is of the form:

p1 (p2) p3

where:

p1 is the lowest frequency desired,

p3 is the highest frequency desired,

p2 is a multiplication constant that is greater than one.

The value for p2 is calculated from the formula:

$$p2 = \left| \frac{p3}{p1} \right|^{1/(k-1)}$$

when k is the total number of frequencys at which a solution is desired. For example: if p1 is A-440 on the musical scale, p3 is A-880, and k = 13; then

$$p2 = (2)^{1/12} = 1.0595$$

and solutions will be produced at the frequencys on the musical scale:

A-440 440 Hz

A# (1.0595)(440) Hz

B $(1.0595)^2(440)$ Hz

C $(1.0595)^3(440)$ Hz

C# $(1.0595)^4(440)$ Hz

... ...

A-880 $(1.0595)^{12}(440) = 880$ Hz

In the above example; thirteen solutions would be produced. ECAP multiplies the preceding frequency by p2 to obtain the new frequency. This process continues until the new frequency exceeds the maximum frequency. For example: if p1 is 1 hz, p3 is 10 hz, and p3 is 2; then solutions would be produced at 1, 2, 4, 8, and 16 hertz. Notice that the last solution frequency is clearly larger than p3. It is <u>important</u> to note that the <u>only</u> difference in form between the two methods for varying frequency is the addition of a + sign before p2 for <u>uniformly</u> spaced increments.

ECAP also provides the user with a 'shortcut' method for varying parameters (other than frequency) over a range in evenly incremented steps. One data group on one card in a problem can have an entry of the form:

p1 (p2) p3

where:

p1 is the initial value of the parameter,

p3 is the final value of the parameter,

p2 is the integer number of uniformed incremented steps of the parameter.

```
/1      7
 //EXAMPLE   JOB (06W12345670001,2,3),URSTUCK,MSGLEVEL=1
 //JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001
 //FT05F001 DD *
 C       EXAMPLE AC ANALYSIS PROGRAM WITH PARAMETER
 C       VARIATIONS
         AC ANALYSIS
 B1      N(0,1), R = 100, I = 10 / 18 (10) 180
 B2      N(1,2),R=.5
 B3      N(2,0), L = 1.0
 B4      N(0,3), L = 2.0
 M1      B(3,4), L = 1.0
 B5      N(4,1), C = 1.0 E -06
 B6      N(4,3), R = .5
 B7      N(0,4), R = .5 E +02
 T1      B(5,7), GM = .1
         FREQUENCY = 66.6
         PRINT, NV , CA , BP
         EXECUTE
 C
         MODIFY
 B1      I = 10 / 45
 T1      GM = .1 ( 9) 1.0
         FREQUENCY = 10
         EXECUTE
 C
         MODIFY
 T1      GM = .5
 M1      L = .1 (10) 1.1
 C       NOTE THAT THE COEFFICIENT OF COUPLING NEVER
 C       EXCEEDS ONE AND THAT THE FREQUENCY = 10 CARRIES
 C       OVER
         EXECUTE
 C
         MODIFY
         FREQUENCY = 20 (1.83224) 4000
 M1      L = 1.0
         EXECUTE
 C
         MODIFY
         FREQUENCY = 65 ( 10 ) 1065
         EXECUTE
         END
 /*
```

Figure 4-7. Example Problem of Figure 4-6 with Variations of Data Parameters.

An example of a 'B' card with the angle of the voltage source incremented would be:

```
/1       7
 B 2     N(3,6), L = 15 , E = 115 / 0 (18) 180
```

and solutions would be produced with the angle of E starting at 0° and increasing in 10° increments to a maximum of 180°. During this angle variation, the rms magnitude of E would remain a constant 115 volts.

Only one parameter can be varied at a time. If frequency is being varied, no other parameter can be varied at the same time. Also, it should be noted that the same type of data group produces a logarithmic variation of frequency and a uniformly incremented variation of any other parameter. If a MODIFY solution follows a parameter variation, the value of the parameter that was varied in the previous solution will be carried over to the MODIFY solution as the last value of the variation. For every case except the logarithmic variation of frequency, the carry-over value would be the p3 value of the variation. To avoid confusion, the proper value of the previously varied parameter should be restated in the MODIFY solution. Figure 4-7 is the example of Figure 4-6 with parameter variations and MODIFY solutions.

It was mentioned earlier that ECAP will not solve circuits where coefficients of coupling are equal to one. This excludes all 'perfect' transformers. There is a technique by which an 'ideal' transformer can be approximated.

An 'ideal' transformer is a device that relates its primary and secondary voltages and currents by the equations:

$$V_s = K\ V_p$$

$$I_s = -I_p\ /\ K$$

where:

K is the 'turns-ratio' of the device,

V_p is the primary 'winding' voltage,

V_s is the secondary 'winding' voltage,

I_p is the primary 'winding' current,

I_s is the secondary 'winding' current.

Since K is a real constant and the power input must equal the power output; an 'ideal' transformer cannot store energy or cause a phase shift. Figure 4-8 shows an 'ideal' transformer with a very small resistance in series with the primary. The device is shown as a transformer, but no actual transformer can ever be an 'ideal' transformer (being lossless excludes this possibility). Let V_p be the voltage as shown in Figure 4-8 instead of the actual primary voltage. This error will be small if the series resistor, R, is small. The modified transformer circuit can be represented by the equations:

$$V_p - R\ I_p = V_s\ /\ K$$

$$I_p = -\ K\ I_s$$

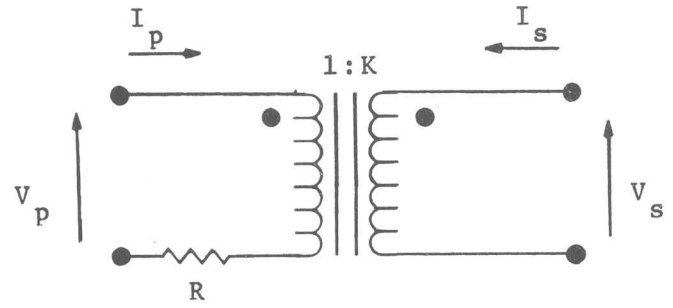


Figure 4-8. 'Ideal' Transformer with a Small Resistor in Series with the Primary Winding

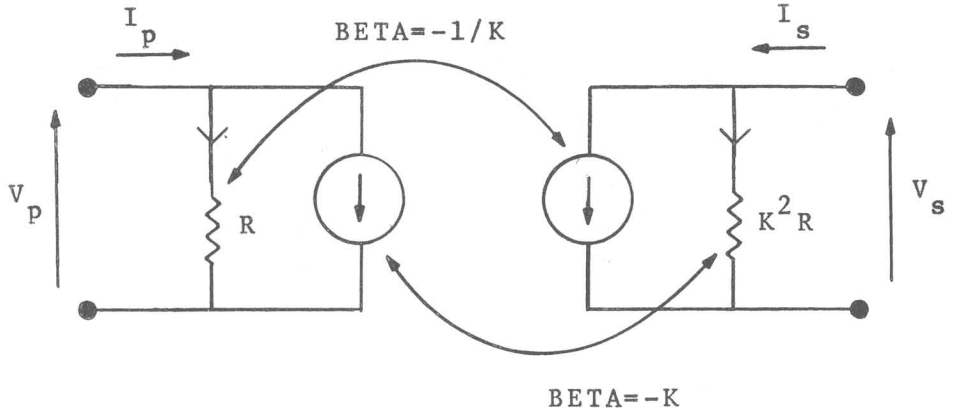


Figure 4-9. Equivalent Circuit for an 'Ideal' Transformer

The preceding equations can be rewritten as:

$$I_p = V_p/R - (V_s/R\ K^2)(K)$$

$$I_s = -\ (1/KR)(V_p) + (1/R\ K^2)(V_s)$$

An equivalent circuit for the 'ideal' transformer can be constructed from these equations and is shown in Figure 4-9. The equivalent circuit is a reasonable representation <u>if</u> the value of R is well below the impedance level of the rest of the circuit. R should be almost a short circuit, but extremely small values could cause inaccuracies in the calculations. The dependent current sources are associated with the branches of resistors R and K^2R. The dependent current source in the secondary has a current gain equal to the inverse of the 'turns-ratio' of the transformer, and is controlled by the current in the small resistance R in the primary. The dependent current source in the primary has a current gain equal to the 'turns-ratio' of the transformer, and is controlled by the current in the small resistance K^2R in the secondary. K^2 is the square of the 'turns-ratio'. Note that the equivalent circuit could also be used in DC circuits as a 'DC Transformer'.

CHAPTER 5

INSTRUCTION GUIDE FOR ECAP/360 TRANSIENT ANALYSIS

Basic Considerations

The response of an electrical circuit to complex waveform excitation can be described by a system of differential equations, but an exact solution of the differential equations may be almost impossible. ECAP approximates a solution to the differential equations by converting them into time-stepped recursion equations. Since the ECAP recursion equations are developed from Kirchoff's law for currents at nodes, the solution of the nodal admittance recursion with present time equal to t_p yields the node voltages at time $t_p + \Delta t$. The term Δt is the time-step of the solution, and is a small positive increment of time. The solution output from ECAP is the approximated node voltages (and other circuit voltages and currents as requested by the user) at integer multiples of the time-step. The ECAP solution is a series of values and not a continuous function.

To simplify the process of specifying a circuit to ECAP and to standardize circuit conventions, a 'Standard Circuit Branch' has been devised. A branch is a resistor, capacitor, or inductor connected between two nodes. A Transient Analysis branch can also have voltage and current sources associated with it. The Transient Analysis 'Standard Circuit Branch' is

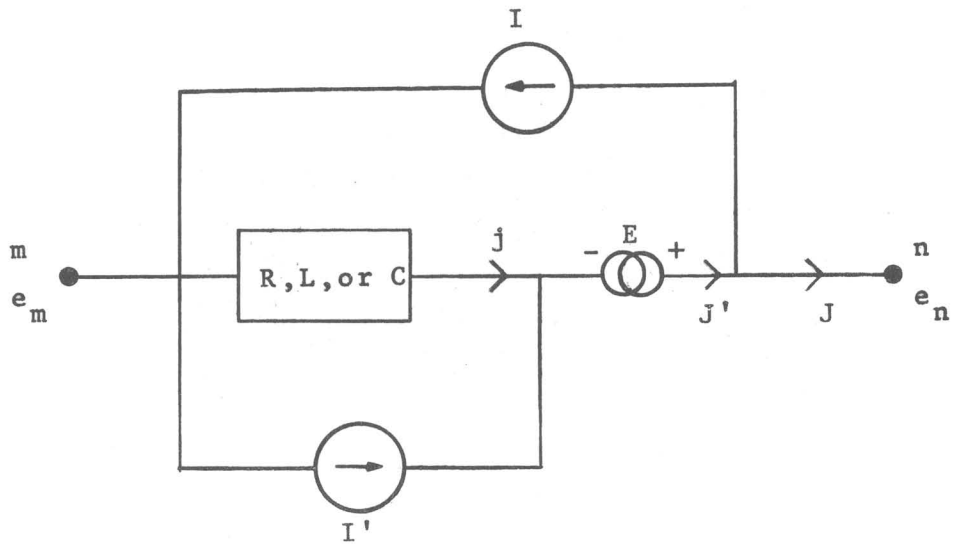


Figure 5-1. ECAP TRANSIENT ANALYSIS 'Standard Circuit Branch'

shown in Fig. 5-1. The terms used in Fig. 5-1 are defined as:

R,L,or C	is the branch element resistance, inductance, or capacitance in ohms, henrys, or farads respectively,
E	is the branch voltage source which can be a fixed DC voltage, a periodic or non-periodic waveform, or both,
I	is the branch current source which can be a fixed DC current, a periodic or non-periodic waveform, or both,
j	is the current flow in the branch element,
I'	is the branch dependent current source (which is controlled by the current flow in the element of another branch),
J'	is defined as the element current,
J	is defined as the branch current,
m	is the 'from' node number of the branch,
n	is the 'to' node number of the branch,
e_m	is the node voltage at node m,
e_n	is the node voltage at node n,
bxx	indicates the branch number.

All positive conventions of the 'Standard Circuit Branch' are shown in Fig. 5-1 with arrows or polarity (+ & -) marks. Positive branch current flows from the 'from' node of the branch. Once the program user specifys the direction of current in the branch, all the conventions of the 'Standard Circuit Branch' apply and the magnitudes of the sources are specified as positive or negative with respect to that convention. Every ECAP branch must have a non-zero value for its resistor, inductor, or capacitor; but the value can be so small as to be almost a

short circuit, or so large as to be almost an open circuit. Those persons who are already familiar with ECAP/360 DC ANALYSIS or AC ANALYSIS will recognize that the 'Standard Circuit Branch' in TRANSIENT ANALYSIS is almost the same, but with the addition of time-dependent sources.

Every node and branch of the circuit must be uniquely numbered. One node must be assigned as the zero-voltage (reference) node. The reference node is assigned the number zero. All node numbers from zero to the highest node number must be assigned. If there are fifteen nodes in a circuit; the highest node number is fourteen, and all numbers between zero and fourteen must be assigned. No node or group of nodes can be isolated (i.e.; without some connection to the reference node). Branch numbers must run from one to the highest branch number. Branch and node numbers are arbitrarily assigned by the program user, as are positive branch current directions. Figure 5-2 shows a circuit with all the convention assignments that would be needed to describe the circuit to ECAP, with the exception of the waveform of the branch sources shown. Notice that the 'Standard Circuit Branch' need not be completely shown for each circuit branch, only the items physically present in the network need be shown.

The maximum size of circuits that ECAP can handle is determined by the internal construction of the ECAP program. Some of the important limits are:

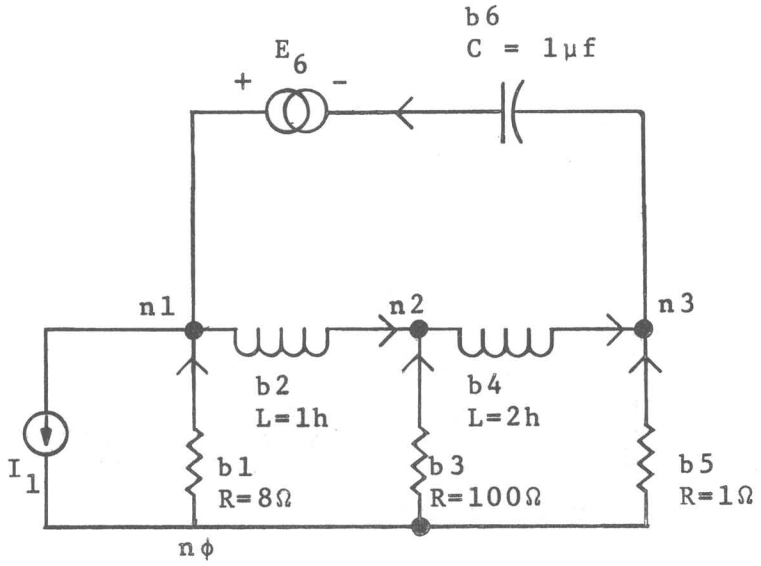


Figure 5-2. Example Transient Circuit with Branch Conventions Assigned

50 nodes, not including the reference node,

200 branches,

200 dependent current sources,

NO dependent voltage sources,

200 switches,

NO mutual inductances.

Since a dependent voltage source in a branch can be converted by the program user into a Norton equivalent dependent current source, the restriction to NO dependent voltage sources is almost trivial. A switch is an ECAP circuit 'trick' that allows the program user to change the value of a circuit parameter during the solution. If the current in the branch that controls the switch changes direction, then some parameter in the branch (branches) controlled by the switch changes value. If a controlled branch is resistive; assigning one value of resistance to be extremely small and the other value to be extremely large approximates the action of a physical switch. Another use of the switch is to approximate non-linear circuit elements. The limit on the number of switches is the maximum number of controlled branches in a circuit. The absence of mutual inductances in ECAP Transient Analysis prevents describing practical transformers to ECAP directly. A technique for describing an 'ideal' transformer to ECAP does exist and will be discussed later. A practical transformer can then be described by the combination of an 'ideal' transformer and a network that represents the winding resistances, inductances, and core.

The dependent current source specified in ECAP is really an isolating amplifier. A dependent current source is physically located in its 'to' (controlled) branch, but its current flow is regulated by the current in the resistor, inductor, or capacitor of its 'from' (controlling) branch. The link between the 'from' and 'to' branch can be specified by a current gain, BETA. BETA must be a real number. If and only if the 'from' branch contains a resistor can the link between the 'from' and 'to' be specified as a transconductance, GM. For those who are familiar with ECAP AC ANALYSIS, note that this condition for specifying dependent current sources is opposite to the condition in AC ANALYSIS. For a resistive branch only; $GM = BETA/R_f$ where R_f is the resistance of the 'from' branch. Figure 5-3 shows the circuit of Figure 5-2 with the addition of two dependent current sources. Notice that a branch may have more than one dependent source.

Table 5-1 shows the definitions of several parameters. Table 5-1 assumes that branch z has 'from' node x, 'to' node y; and the dependent source in branch z has 'from' branch q and current gain $BETA_{qz}$. In Table 5-1, it is important to note the definition of branch current, element current, and instantaneous branch power. Specifically, it is the method of calculating instantaneous branch power that justifies the Figure 5-1 representation of the 'Standard Circuit Branch'. The 'Standard Circuit Branch' of Figure 5-1 (which applies to ECAP/360)

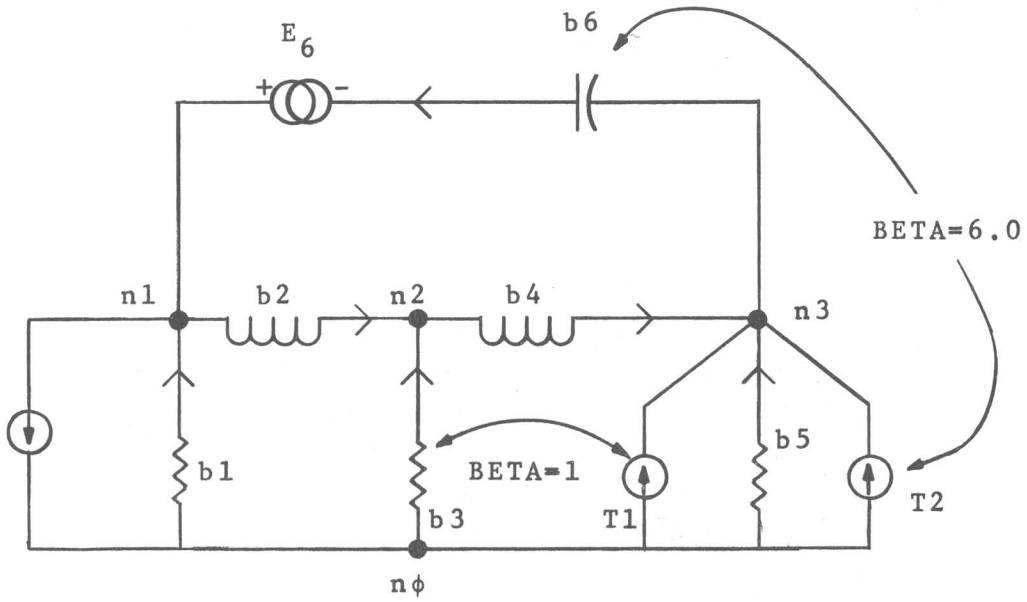


Figure 5-3. Example Circuit of Figure 5-2 with Two Dependent Sources Added

Parameter	Symbol	Defining Equation	Available as an ECAP/360 Output	Output 'PRINT' Code
Node x Voltage	e_x	Solution of System Eq.	YES	NV
Branch z Voltage Source	E_z	User Data	NO	
Branch z Voltage	V_z	$V_z=e_x - e_y$	YES	BV
Element Voltage	V'_z	$V'_z=V_z + E_z$	YES	CV
Branch z Current Source	I_z	User Data	NO	
Branch z Dependent Source	I'_z	$I'_z= GM_{qz}V_q$	NO	
Branch z Impedance	Z_z	Computed from User Data	NO	
Impedance Current	j_z	$j_z=V'_z/Z_z$	NO	
Element Current	J'_z	$J'_z=j_z + I'_z$	YES	CA
Instantaneous Branch z Power	P_z	$P_z=J'_z\ V'_z$	YES	BP
Branch z Current	J_z	$J_z= J'_z - I_z$	YES	BA

Table 5-1. ECAP TRANSIENT ANALYSIS definitions for Parameters

is not the same as the one shown in the IBM ECAP/1620 USER'S MANUAL; though this manual in general applies to ECAP/360. Since inductors and capacitors store energy, the instantaneous branch power for branches with inductors and capacitors will not in general be zero.

Numerical data can be represented to ECAP with up to eight digit, with or without a decimal, and with or without an E followed by an integer that indicates an exponential power of ten. The largest number accepted by ECAP is 7 E +75, which is 7 followed by 75 zeros. The exponent can also be a negative integer. Sometimes the program output used D instead of E, but it means the same thing. Using D on an input data card to indicate an exponent is not allowed.

Basic Instructions Required to use ECAP/360 Transient Analysis

ECAP/360 at the present is available from disc storage at the KSU Computing Center. Instructions to the computer can be broken into eight classifications:

1. Commands to get the program from storage (5 cards),
2. User circuit data,
3. Solution Controls,
4. Output Specifications,
5. Commands,
6. System Controls,
7. Comments,
8. Termination of job instructions (2 cards).

All input to the computer is by punched cards, and all output from the computer is printed. There is no punched card output option. In the remainder of these instructions; the standard eighty entry data card will be represented by a /1 above the position where column one would be, and a 7 above column seven. Imbedded blanks (an imbedded blank is a blank entry between two other characters) are assumed to be one character entry long unless a column number entry is shown above the character following the blanks. No entrys are allowed in columns 73 - 80. A sample of two comment cards (one after the other) by this system of representation would be:

```
/1      7
 C      TRANSIENT ANALYSIS TEST PROGRAM
 C      BUZZ-SAW WAVE EXCITATION INTO FILTER
```

There are five special cards required to get the program from the disc. They are <u>not</u> part of ECAP, but are control cards required by the Computing Center for orderly operation. All five of them <u>must</u> be punched in the EBCDIC (extended binary code) character set. The EBCDIC characters are the ones punched by an IBM 29 Card Punch, or multiple punched on an IBM 26 Printing Card Punch. The 26 Printing Card Punch normally punches BCD (binary code) characters. The first card must be a 'job card', exactly the same as for other programs submitted to the Computing Center. At the present, this card is completely specified in KSU Computing Center Notice #52, dated 27 February 1968. An abbreviated form of this card which 'will work' is:

```
                1   1
/1              2   6
 //jobname      JOB (pano,time,lines),yername,MSGLEVEL=1
```

where:

// must appear in columns 1 and 2,

jobname is 1 - 8 characters which name the job, the first of which must be alphabetic,

JOB must start in column 12,

(must appear in column 16,

pano is your job charge number,

time is the maximum running time in integer minutes,

lines is the maximum number of lines of printed output in integer thousands of lines,

yername is your own name with no imbedded blanks.

A sample of a job card would be (with a false job number):

```
                1   1
/1              2   6
 //ALLERROR JOB (06W190978Q001),5,4),URSTUCK,MSGLEVEL=1
```

If the running time of the problem exceeds the 'time' entry on the job card; the computer will automatically terminate the job. The job will also be terminated if the number of 'lines' is exceeded. Either the 'time' or 'lines' entry (or both) can be omitted. If 'time' is omitted, the data group inside the parentheses becomes (pano,,lines). There can be no blanks between the commas. The computer will assume a 'time' entry of 1 minute. If the 'lines' entry is omitted, the data group inside the parentheses becomes (pano,time). The computer will assume a 'lines' entry of 1. If both 'time' and 'lines' are omitted; the entry becomes (pano), and the assumed entrys are the same.

The remainder of the five special command cards as presently specified are:

```
/1      7
 //JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
 //FT05F001 DD *
```

These five special cards must be the first five cards in any ECAP problem deck, and they must be in the order presented.

Two special cards are used to terminate an ECAP job, and they are the last two cards in an ECAP problem deck. In proper order, they are:

```
/1      7
        END
 /*
```

Comment cards can be placed anywhere after the first five cards and before the last two cards of the problem deck. Column 1 of a comment card must have the character C punched in it. Any remark can be included in columns 2 - 72. Two examples of comment cards were included as an earlier example.

Four types of data cards are used to describe the problem circuit to ECAP. The first type of data card specifies all the data from one branch, except the dependent source and time-dependent source data. This card is called a 'branch' or just 'B' card. The standard forms for the 'B' card are: for resistive elements,

```
/1      7
 Bxx    N(ww,zz),R=rr,E=ee,I=ii
```

for inductive elements,

```
/1      7
 Bxx    N(ww,zz),L=ll,E=ee,I=ii,IO=io
```

for capacitive elements,

```
/1      7
 Bxx    N(ww,zz),C=cc,E=ee,I=ii,EO=eo
```

where:

B indicates that it is a 'B' card,

xx is the number of the branch, and can appear in any of the columns 2 - 5,

N indicates the node connection of the branch,

ww is the 'from' node of the branch,

zz is the 'to' node of the branch,

rr is the resistance of the branch in ohms (if it is a resistive branch),

ll is the inductance of the branch in henrys (if it is an inductive branch),

cc is the capacitance of the branch in farads(if it is a capacitive branch),

ee is the magnitude in volts of a fixed-magnitude DC voltage source in the branch,

ii is the magnitude in amperes of a fixed-magnitude DC current source in the branch,

io is the initial current in the inductor of an inductive branch at the time the solution starts,

eo is the initial voltage on the capacitor of a capacitive branch at the time the solution starts.

The entrys rr, cc, and ll cannot be zero; but they can be very small, or even negative. If the conductance of a resistive branch is known; the entry R=rr can be replaced with G=gg, where gg is the conductance of the resistor in mhos. If any

data is not entered on a data card, ECAP will assume that data to be zero. For example; if EO=eo is not entered on a capacitive branch 'B' card, ECAP assumes the initial voltage on the capacitor is zero. Some typical 'B' cards are:

```
/1      7
 B 2    N(0,12), R = .12345678 E -02
 B17    N(32,8) , L = .11 , I = 3
 B33    N(4,6)  , C = 10 E -06, EO = 66
 B21    N(6,4) ,  L = 12.8 , I = 5 , E =-88 , IO = -3
```

The second type of data card specifies dependent current source data. It is called a 'T' card after its column one entry. The standard form of a 'T' card is

```
/1      7
 Txx    B(bb,dd),BETA=tt
```

or

```
/1      7
 Txx    B(bb,dd),GM=kk
```

where

T indicates a dependent source, 'T', card,

xx is the number of the dependent source,

bb is the 'from' branch of the dependent source,

dd is the 'to' branch of the dependent source,

tt is the current gain of the dependent source,

kk is the transconductance of the dependent source (only if the 'from' branch is resistive).

Only one 'T' card is needed for each dependent source. In general, the 'from' branch element can be an inductor, capacitor, or resistor. The dependent source can be specified with a GM only if the 'from' branch is resistive; but it can be specified with a BETA for all cases. Those persons who

are already familiar with ECAP/360 AC Analysis should recognize that this dependent source specification condition for Transient Analysis is opposite to that of AC Analysis. In the case of a resistive 'from' branch, GM and BETA are related by the formula GM = BETA/R_c where R_c is the resistance of the 'from' branch. ECAP restricts BETA (and GM when used) to real numbers. The dependent sources are uniquely numbered in order from one to the highest number. No number between one and the highest can be missing. Some typical 'T' cards are:

```
/1      7
 T 1    B(6,3), BETA = 2.33
 T 7    B(.23,200), BETA = 0 .456 E +01
 T 8    B(1,2), GM = .00118
```

The third type of data cardspecifies time-dependent source data. There are three variations of the time-dependent source cards: non-periodic, periodic, and sine wave. On the periodic or non-periodic cards; the program user provides the value of the waveform at up to a maximum of 126 evenly time spaced increments, and ECAP linearly approximates the value of the function between the provided data. The increment of time between data points must be an positive integer multiple of the specified time-step. Time-step and time-step selection will be discussed later. Sine wave time-dependent source data is automatically generated from trigonometric tables internally available to ECAP.

The standard form of a non-periodic time-dependent source card is: for time-dependent voltages,

```
/1      7
 Exx    (k), p_o , p_1 , p_2 , ..., p_j,..., p_n
```

or for time-dependent currents,

```
/1      7
 Ixx    (k), p_o , p_1 , p_2 , ...,p_j,..., p_n
```

where:

E indicates the time-dependent source is a voltage source,

I indicates the time-dependent source is a current source,

xx is the branch number of the branch in which the time-dependent source is located,

k is the integer number of time-steps between data points,

p_o is the initial value of the time-dependent source at the time the solution starts,

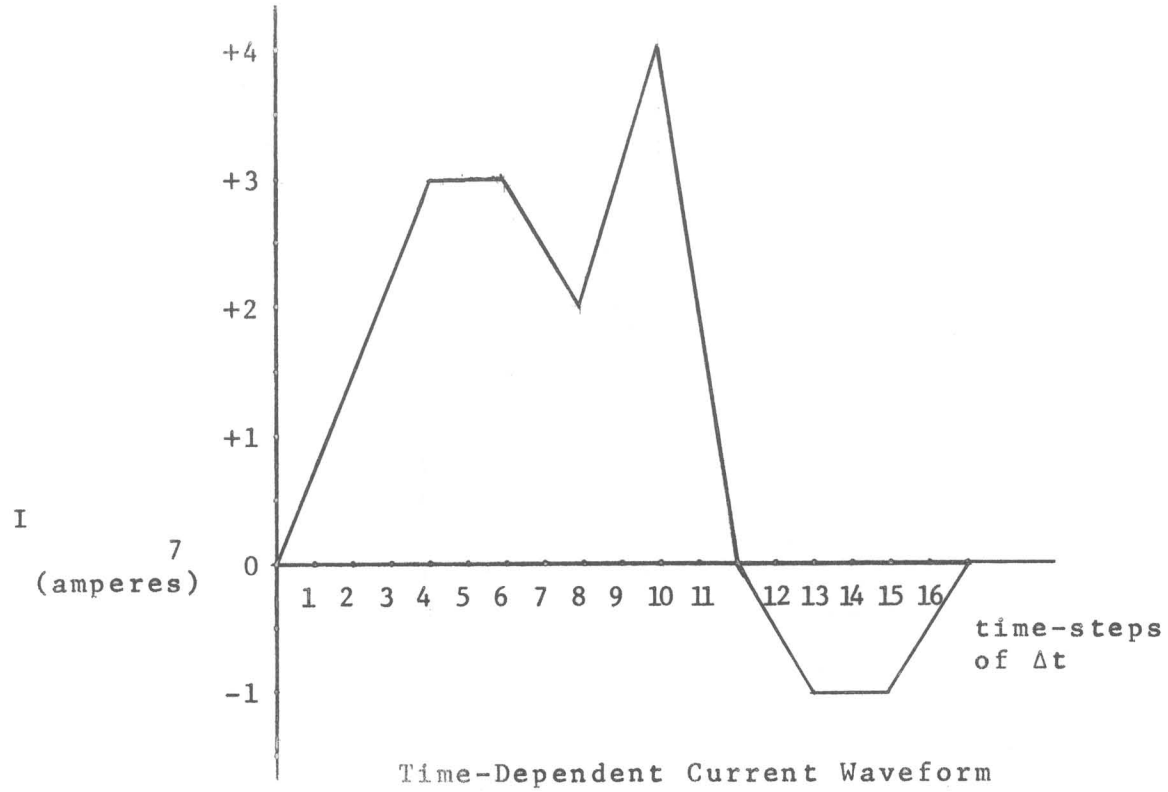
p_j is the value of the time-dependent source at the j'th increment of time, [time = $(jk)\Delta t$],

p_n is the final value of the time-dependent source,

If ECAP is allowed continue solving for circuit response past the time of the last entry, p_n, on a non-periodic time-dependent source card; ECAP holds the value of p_n for the remainder of the solution time. Figure 5-4 shows a typical example of waveform generation with a time-dependent source card.

The standard form of a periodic time-dependent source card is almost the same as the non-periodic source card; the only difference is the character P in column seven of the card. For a periodic time-dependent voltage source, the card would be of the form

```
/1      7
 Exx    P(k), p_o , p_1 , p_2 , ... , p_j , ... , p_n
```



Time-Dependent Current Waveform

Sample interval k = 2 time-steps, as shown with large dots on function plot.

Time-Dependent Source Card for the Waveform.

```
/1      7
 I 7    (2), 0 , 1.5 , 3 , 3 , 2 , 4 , 0 ,-1,-1, 0
```

Figure 5-4. Typical Example of Time-Dependent Source Waveform Specification.

All definitions from the non-periodic time-dependent source card carry over to the periodic card. If ECAP is allowed to continue solving for circuit response past the time of the last entry, p_n, on the source card; ECAP repeats the function starting again from p_1. For periodic sources, p_o and p_n must be the same value. It should be noted that since ECAP linearly approximates a time-dependent source between specified data points; a time-dependent waveform with a discontinuity cannot be described. The best approximation of a discontinuous waveform would have a finite rise time at the discontinuity of k multiples of the time step, where k is the time increment of the source as specified on the source card.

The standard form for a sine wave time-dependent source is: for voltage sources,

```
/1      7
 Exx    SIN( tp) , Vp , Vb , to
```

or for current sources,

```
/1      7
 Ixx    SIN( tp ), Ip , Ib , to
```

where:

SIN indicates that the source is a sine wave,

t_p is the period of the sine wave in seconds,

V_p is the voltage from average to peak,

V_b is a DC offset or bias voltage,

I_p is the current from average to peak,

I_b is a DC offset or bias current,

t_o is a time offset (changes phase relation).

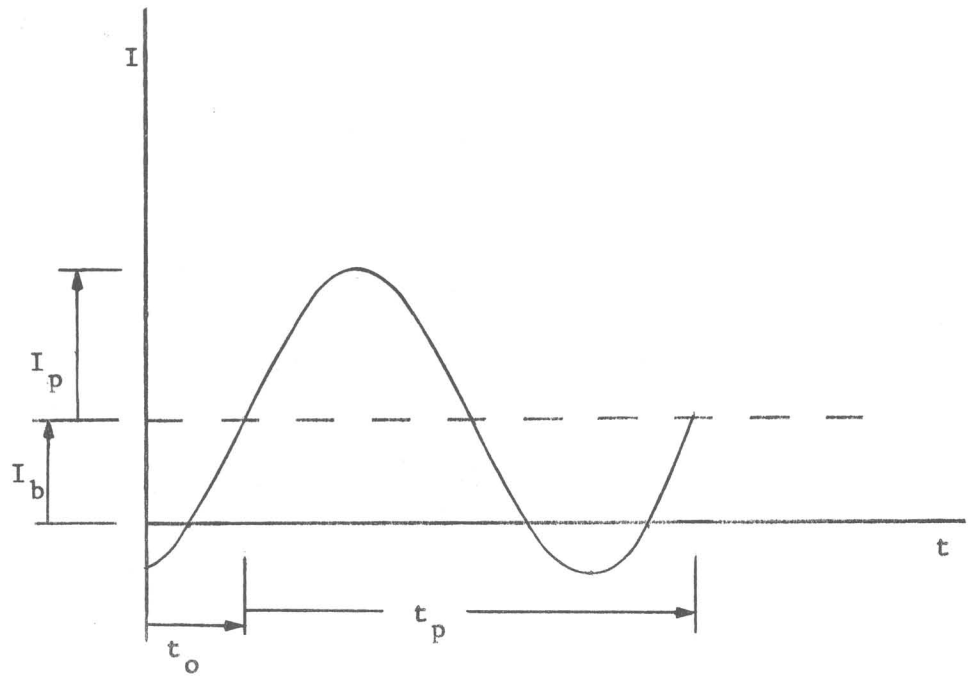


Figure 5-5. Pictorial Representation of Data Specified on SIN Time-Dependent Current Source Cards

Figure 5-5 shows an example of a sine wave current with all parameters needed for specifying the source card shown. The parameter t_p is the inverse of the wave frequency. The maximum value of t_p is 126 times the problem time-step. The minimum value of t_p is 3 times the problem time-step.

All the time-dependent source cards are really just extentions of the E and I entrys on the 'B' card. If E or I (a DC bias) is entered in the 'B' card, the information from the time-dependent source card is added to it. In the case of the SIN source cards, the DC bias can and should be entered only on the SIN card to prevent confusion. Also, no more than one E and one I time-dependent source card can be included for each circuit branch.

The space provided on a 'B' card or a periodic (non-periodic) source card can easily be exhausted. By placing an * in column six, the data can be continued on the next card (cards). No other type of card can be continued. An example of a continued 'B' card is:

```
/1    67
 B 3   N(1,2), L = 1.5
       *       E = - 6 E +01
       *       IO = 15
```

Note that the continuation cards contain no entrys to identify which branch 'B' or time-dependent source card is being continued. The program user should mark the continuation cards with the number of the continued card, in ink or pencil to eliminate possible confusion. And to repeat, a continuation card must follow immediately in the deck the card it is to continue.

A switch as mentioned before is an ECAP 'trick' for changing the value of a circuit element during the execution of a solution. A switch can be either ON or OFF. If the current through the element of the controlling branch is positive, the switch is ON. If that current is negative, the switch is OFF. If the current changes from positive to negative, the switch goes from ON to OFF. If the current changes from positive to zero, the switch is still turned OFF. Just the opposite holds true for changes from negative to positive. The program user 'guesses' whether the initial current in a branch is going to be positive or negative, and from this guess specifys the initial condition of the switch as either OFF or ON. When a switch actuates, it changes the values of parameters in all branches associated with the switch. An example here is more meaningful than words.

```
/1      7
 S 1    B = 6, ( 1, 3 , 6 , 14 ), OFF
 B 1    N(1,2), R = ( 1 E +07 , 2 E -02 )
 B 3    N(2,3), L = 8 , I = ( + 2 , - 2 ) , E = ( 0 , 3 )
 B 6    N(3,4), C = 1 E -06 , I = 3 , E = (1,2), EO=5
 B14    N(4,5), C = 5 E -07
```

In the example above, switch number one is controlled by the current in the capacitor of branch six. It is guessed that the initial current will be negative, as is seen by the entry on card S1 : OFF. On all the branch cards associated with switch number one (branches 1, 3, 6, & 14), the first value entered in the parenthesis is the value of that parameter associated with the initial condition of the switch. The initial condition is

OFF, so the value of the resistor in branch number one (B 1 card) is ten megohms. If the current in the capacitor of branch number six changes to positive; the switch is turned ON, and the last value entered in the parenthesis is substituted for the first value. The ON value of the resistor in branch number one is twenty milliohms. Branch one acts almost like a contactor between nodes one and two. From the other 'B' cards of the example it can be seen that inductance and capacitance values can be switched too. The value of the DC voltage and current bias in a branch can also be switched, but EO and IO values cannot be switched (it wouldn't be meaningful if they could). Note that branch six switches itself. Branch fourteen won't change at all since no ON and OFF alternative values are given. The entrys on 'T' and time-dependent source cards can <u>not</u> be switched. If in the example the initial condition had been specified as ON, then the first entrys inside a parenthesis would be associated with ON rather than OFF.

From the preceding discussion it can be seen that the standard form of a switch card is:

```
/1      7
 Sxx    B = bb ,( b1 , b2 , ... , bj ), OFF
```

or

```
/1      7
 Sxx    B = bb , ( b1 , b2 , ... , bj ), ON
```

where:

S indicates that the card contains switch information,

xx is the number of the switch,

bb is the branch that controls the switch (the 'relay coil'),

b_j is a branch number of a branch that is controlled by the switch,

OFF
ON are initial condition specifications for the switch where positive initial current in the element of the control branch is ON and negative initial current is OFF.

In branches that are controlled by the switch, the data entrys are of the form:

$$(D_i , D_s)$$

where:

D_i is the initial value of the parameter,

D_s is the switched value of the parameter.

Switch numbers must be assigned in order from one to the highest number needed. No switch number can be skipped. As would be expected, a branch can be switched by only one switch. A method for getting around this last restriction is to series or parallel connect several branches and have each switch control a 'separate' branch. Using this method, non-linear circuit elements and electronic components can be approximated as piecewise linear elements. The IBM ECAP/1620 USER'S MANUAL devotes many pages to approximating non-linear elements, and all of its discussion applys to ECAP/360. Since the program user must guess the initial condition of a switch, it is possible to be wrong. ECAP will change the sense of the switch for its first time step solution, but the initial condition solution will be with all switches as specified. This may

cause errors in the solution; so if a switch actuates in the first solution the sense of switch should changed on its 'S' card, and all data in parenthesis on the controlled branches 'B' cards should be reversed. Since approximating non-linear elements requires many switches, a person should be very familiar with ECAP Transient Analysis solutions and trouble-shooting before attempting any complex approximations.

Usually, the solution of a transient problem starts at time zero. In the absence of instructions to the contrary, ECAP assumes the initial time to be zero. The program user can specify a different initial time by including the card:

```
/1      7
        INITIAL TIME = t_o
```

where t_o is the initial time desired in seconds. Positive and negative numbers can be used for t_o. The user must specify a final time in seconds, and the card for this specification is:

```
/1      7
        FINAL TIME = t_f
```

where t_f is the proper final time. Computer time used for the solution of a transient problem is almost a linear function of the final time, so final time specification should be as small as necessity will allow. As a protection against excessive time, the maximum time specification on the 'job card' will over-ride the transient solution and terminate the job before its completion.

Since ECAP solves transient problems by repeated solutions of a set of recursion equations based on a time-step Δt, selection of a proper time-step is extremely important to the program user. If the time-step is too short, excessive computer time will be required. If the time-step is too large, the approximations involved in the solution may be too coarse to maintain the desired accuracy. If somehow a 'time constant' could be assigned to the whole problem circuit, then a time-step of one tenth of the time constant would be a reasonable selection. The accuracy of the solution could be increased by reducing the time-step, but quite soon the 'law of diminishing returns' would preclude further reduction.. One method of time-step selection is based on that law. Usually the most violent transients will occur shortly after the initial time or after a switch actuation. Using this 'fact', the program user specifys a time-step that he knows is too large for an accurate solution and a final time that is just long enough to get a solution for the most violent transient. The time-step is then reduced by a factor of ten with the same final time. A comparison of solutions will indicate the improvement in accuracy. After a few reductions of the time-step, the solution will be accurate enough. Another method for estimating the proper time-step requires the calculation of time constants for pairs of elements connected to the same node. These time constants are calculated as if the rest of the elements in the circuit did not exist, therefore they are called 'isolated time constants'. If for example a 1000 ohm resistor and a 10 microfarad

capacitor were connected to the same node, the isolated time constant would be RC = .01 seconds. Obviously all pairs of resistors, pairs of capacitors, and pairs of inductors can be ignored since they have no time constant. Since the time constants (or time periods) of interest are computed from the formulas:

R/L for resistor-inductor pairs,

RC for resistor-capacitor pairs,

$\sqrt{LC}$ for inductor-capacitor pairs

only the smallest resistance and capacitance at each node need be considered. Also, only the largest and smallest inductor at each node need be considered. After locating the smallest 'isolated time constant', a reasonable selection of a time-step is one tenth of the smallest 'isolated time constant'. It should be remembered that in any circuit with a time-dependent source a change in time-step will change the 'frequency' of the source. Therefore; if the time-step is to be refined, new time-dependent source cards should be included with appropriate corrections for the difference in time-step. Also, ECAP will accept almost any positive number as a valid time-step; but it will punish the program user who specifys a time-step like .008375 seconds by producing solutions in multiples of that time-step. It is much more convenient to use time-steps that are integer powers of ten. The program user must provide the computer with the time-step information; on a card of the form:

```
/1      7
        TIME STEP = t_s
```

where t_s is the time step in seconds.

The exact form and amount of solution data produced by the computer is controlled by two cards. The first is a solution control card of the form:

```
/1      7
        OUTPUT INTERVAL = n
```

where n is the number of time-steps between printed outputs. The number n must be a positive integer. This card allows the program user to improve the accuracy of the calculations without being buried alive in a blizzard of output data. Since the output data is in tabulated form, the output interval may be dictated by the number of data points the program user is willing to plot on a graph. For extremely violent transients, requesting an output for every time-step may be necessary. If the number of lines of printed output exceeds the 'lines' entry on the 'job card', the job will be automatically terminated. The second card that controls the solution output is an output specification card of the form:

```
/1      7
        PRINT , xx , xx , xx , xx
```

where xx indicates two letter output code for solution options. Table 5-2 shows the solution options available from ECAP Transient Analysis. It may be helpful to refer again to Table 5-1 which properly defines the circuit voltages and currents. Some typical PRINT cards would be:

```
/1      7
        PRINT , NV
        PRINT , NV, BV , CA
        PRINT , BP
        PRINT , NV , BV , CV , CA , BA , BP
```

Only one PRINT card and one OUTPUT INTERVAL card is needed for each problem.

Code	ECAP Solution Output
NV	NODE VOLTAGES
BV	BRANCH VOLTAGES
CV	ELEMENT VOLTAGES
CA	ELEMENT CURRENTS
BA	BRANCH CURRENTS
BP	INSTANTANEOUS BRANCH POWER

Table 5-2. Solution Output Codes

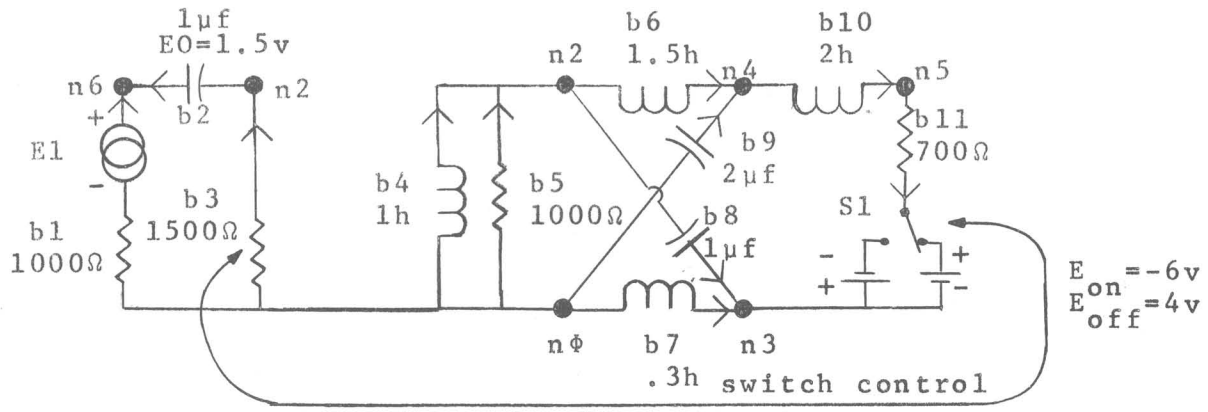
Only two additional cards not already discussed are necessary before a complete problem can be described to the computer. One card commands ECAP to:

```
/1      7
        EXECUTE
```

the solution. The EXECUTE card is placed after the data in the problem deck. The other card is:

```
/1      7
        TRANSIENT ANALYSIS
```

which specifys to the computer that it is handling a Transient problem. The TRANSIENT ANALYSIS card is placed just ahead of the data in a problem deck. A complete example problem is shown in Figure 5-6. Notice that the TRANSIENT ANALYSIS card comes after the five initial cards required by the Computing Center to get the program from disc, and the EXECUTE card comes just before the END and /* cards which terminate the problem deck.



E1 has the same waveform as example of Figure 5-4 with k changed to 20, and periodic.
S1 assumed to be ON initially.
Time-step selected as .001 seconds.

```
Card
Classification
Number                          Program Cards
                        1
      /1     7          2
  1    //EXAMPLE  JOB (06W1234567Q001,2,2),U_R_STUCK,MSGLEVEL=1
  1    //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
  1    //STEP1 EXEC PGM=ECAP
  1    //FT06F001 DD SYSOUT=A,DCB=RECFM=UA
  1    //FT05F001 DD *
  7    C      EXAMPLE TRANSIENT PROBLEM
  5           TRANSIENT ANALYSIS
  3           2ERROR = .1
  3           3ERROR = .001
  3           INITIAL TIME = 0.0
  3           TIME STEP = .0001
  3           OUTPUT INTERVAL = 5
  3           FINAL TIME = .050
  2    B1     N(0,6) , R = 1000
  2    E1     P(20),0,1.5,3,3,2,4,0,-1,-1,0
  2    S1     B = 3 , (11) , ON
  2    B2     N(1,6), C = 1E-06
  2    B3     N(0,1), R = 1500
  2    B4     N(0,2), L = 1
  2    B5     N(0,2), R = 1000
  2    B6     N(2,4), L = 1.5
  2    B7     N(0,3), L = .3
  2    B8     N(2,3), C = 1E-06
  2    B9     N(0,4), C = 2E-06
  2    B10    N(4,5), L = 2
  2    B11    N(5,0), R = 700 , E = (-6,4)
  4           PRINT, NV
  5           EXECUTE
  8           END
  8    /*
```

Figure 5-6. Example Circuit and Problem Deck for ECAP/360 TRANSIENT ANALYSIS

The cards between TRANSIENT ANALYSIS and EXECUTE can be shuffled without effecting the operation of the program. The only exception would be when the problem deck includes 'B' card or time-dependent source card continuations.

Additional ECAP TRANSIENT ANALYSIS Solution Features

ECAP Transient Analysis has three built in features that maintain the accuracy of solutions. The first feature computes the sum of the currents at each node. By Kirchoff's nodal law the sum of node currents should be zero; but ECAP is limited to sixteen significant figures accuracy (eight in input and output) in its calculations, so node current sums will be small but not zero. After computing the difference from zero at each node, ECAP sums the absolute value of these differences at all nodes. This sum is called the 'Current Unbalance' of the solution. If the current unbalance in the solution exceeds .001 ampere more than twenty times during the execution of a transient problem; ECAP will terminate the problem immediately after the twenty first excessive unbalance, and an error message will be printed. Current unbalances can be caused by a poorly chosen time-step or extremely large differences between component values. Usually component values can differ by several powers of ten before accuracy problems are encountered. Sometimes it may be desirable to change the maximum allowable current unbalance in the circuit. The card:

```
/1      7
        1ERROR = xx
```

where xx is the desired maximum current unbalance in amperes. If a 1ERROR card is not entered, the program will use .001 amperes as the maximum current unbalance.

The second accuracy maintaining feature allows the program to pinpoint the exact actuation time of switches in the circuit. This feature is needed because the exact actuation time may occur between two time-step solutions. If the program user does not specify otherwise, ECAP will locate the actual actuation time to within one thousandth (.001) of a time step. If the time-step were one millisecond, the actuation time would be accurate to one microsecond. For some transient problems, this may be more or less accuracy than is needed. The resolution is changed by the card:

```
/1      7
        2ERROR = xx
```

where xx is the desired resolution. The entry must be positive and less than one. For example; if xx were .1, switch actuation time resolution would be within .1 (10%) of a time-step. It should be noted that ECAP will produce two solutions at the exact time of switch actuation; one for time immediately before the actuation, and the other immediately after.

The third accuracy maintaining feature allows the program user to automatically reduce the time-step after every switch actuation. Since the more violent transients usually occur after a switch actuation, this feature protects the accuracy of the solution without the penalty of an extremely short time-step. An example will serve to best describe this feature. Assume

that a switch actuation time has been located between two normal time-steps exactly 23.5% of a time-step past the last full time-step solution. There is now 76.5% of a time-step left between the actuation time and the time of the next full time-step solution. Call this remainder of the interval R. If the card:

```
/1      7
        3ERROR = p
```

where p is a positive decimal between zero and one. ECAP then computes four new reduced time-steps to fill out the remainder of the interval. The first reduced time-step after the actuation will be pR. The second reduced time-step will be $(p^{2/3}-p)R$. The third reduced time-step will be $(p^{1/3}-p^{2/3})R$. The fourth reduced time-step will be $(1-p^{1/3})R$. The sum of these reduced time-steps is:

$$[p + p^{2/3} - p + p^{1/3} - p^{2/3} + 1 - p^{1/3}]R = R$$

After ECAP fills out the remainder of the interval, it returns to the normal user specified time-step until the next switch actuation. If p were chosen as .001; the reduced time-steps would be (in order): .001R, .009R, .09R, and .9R. If the program user does not specify p on the 3ERROR card, ECAP will not reduce the time-step for increased accuracy. Instead, it will use R as the next time-step and then return to the normal time-step.

If a transient problem has no periodic sources, it is reasonable to expect the transients to fade away and leave a

steady state or equilibrium condition. Even a network with only practical inductors and capacitors (lossy elements) will eventually reach an equilibrium. If the FINAL TIME of the solution were set equal to 7 E +75 seconds; this might yield an equilibrium solution, but the cost in computer time would be fantastic. ECAP offers another 'short-cut' when the card:

```
/1      7
        EQUILIBRIUM
```

is included in the problem deck somewhere between TRANSIENT ANALYSIS and EXECUTE. After finishing the transient analysis from the initial time to the final time; ECAP substitutes an almost short circuiting resistance for all inductors and an 'almost infinite' resistance for all capacitors, and then executes a DC circuit solution. In the absence of user instructions, ECAP assumes the short circuiting resistances to be .01 ohms. The user can specify a different shorting resistance if the card:

```
/1      7
        SHORT = ss
```

where ss is the desired shorting resistance in ohms. In the absence of user instructions, ECAP assumes the 'almost infinite' resistance to be ten megohms. The user can specify a different shorting resistance if the card:

```
/1      7
        OPEN = oo
```

where oo is the desired value of the 'almost infinite' resistance in ohms.

When the program solves for an 'initial condition', it also substitutes the values of OPEN and SHORT into the circuit. Thus, the voltage across a capacitive branch that had no initial voltage specified will not be zero at time zero (the first ECAP solution); instead it will be some very small value. Likewise, the current in an inductor that had no initial current flow will be some small value at time zero. Examination of the time zero solution will reveal if the values of OPEN and SHORT are appropriate. ECAP provides two solutions everytime it locates (within 2ERROR) a switch actuation time. The first solution is at the actuation time with the conditions as they are before the switch changes. The second solution is an 'initial condition' solution with conditions as they are after the switch changes, but with OPEN and SHORT substituted for inductors and capacitors.

Sometimes it is inconvenient to punch cards with EBCDIC characters. If a group of cards (other than the first five cards in the problem deck that get the program from disc storage) have been punched with BCD characters; they can be used if the card:

```
/1      7
 *BCD
```

is inserted in the deck immediately ahead of the BCD punched group of cards. If later in the deck EBCDIC punched cards are included again, the card:

```
/1      7
 *EBCDIC
```

converts the reading interpretation back to EBCDIC characters. In the absence of instructions, ECAP assumes all cards are punched with EBCDIC characters. If this assumption is wrong, a battery of error messages will be produced. The error messages will indicate by card number (in the order they appear in the deck, excluding all comment cards and the first five cards) the card (cards) in error. The error message will also pinpoint the column of that card where an incorrect character occurs.

ECAP/360 was tailored to take advantage of the improved calculating speed of IBM's System 360. The program user is given the option of loading problems in batches by placing another problem after the EXECUTE card of the previous problem. The 'new' problem must have its own TRANSIENT ANALYSIS card as its first card and its own EXECUTE card as its final card. The data between the TRANSIENT ANALYSIS and EXECUTE cards of the 'new' problem must be complete. Those persons who are familiar with ECAP DC and AC Analysis should realize that this is a definite difference and the 'short-cut' methods available in AC and DC analysis do not apply to Transient Analysis. Several problems can be batched on one job card, and the END and /* cards are placed after the last problems EXECUTE card. Figure 5-7 shows an example of a batch of problems, though the actual data cards have been omitted. The card classifications and card formats are reproduced in Table 5-3 and Table 5-3 (cont.). Figure 5-6 is an example of converting a transient circuit into an ECAP/360 TRANSIENT ANALYSIS program deck. Figure 5-6 includes

Card Class. Number	Card Class.	Card Name	Card Format
			/1 7
1	Commands to get the Program	jobcard	(as currently specified)
		None	//JOBLIB DD DSNAME=SYS1,USERLIB,DISP=OLD
		None	//STEP1 EXEC PGM=ECAP
		None	//FT06F001 DD SYSOUT=A,DCB=RECFM=OLD
		None	//FT05F001 DD *
2	User's Circuit Data	'B'	Bxx N(y,z), R=r , I=i , E=e
		'B'	Bxx N(y,z), L=1 , I=i , E=e
		'B'	Bxx N(y,z), C=c , I=i , E=e
		'E'	Eaa (k),a,b,c,d,.....,h
		'E'	Eaa P(k),a,b,c,d,.....,h
		'I'	Iaa (k),a,b,c,d,.....,h
		'I'	Iaa P(k),a,b,c,d,.....,h
		Continuation	* Remaining data of a 'B','E', * or 'I' card of the above types
		'E'	Eaa SIN(t),vp,vb,to
		'I'	Iaa SIN(t),ip,ib,to
		'S'	Snn B=w , (b,c,z,...,q), ON
		'S'	Snn B=w , (b,c,z,...,q), OFF
		'T'	Tqq B(a,b), BETA = mm
		'T'	Tqq B(a,b), GM = gg
			Note: GM can be used only if 'from' branch is resistive
3	Solution Control	None	TIME STEP = p
		None	INITIAL TIME = to
		None	FINAL TIME = tf
		None	SHORT = xx
		None	OPEN = zz
		None	EQUILIBRIUM
		None	1ERROR = aa
		None	2ERROR = bb
		None	3ERROR = cc
		None	OUTPUT INTERVAL = k
4	Output Spec.	'Print'	PRINT,NV,BV,CV,CA,BA,BP

Table 5-3. ECAP/360 TRANSIENT ANALYSIS Input Cards

Card Class. Number	Card Class.	Card Name	Card Format
5	Commands	None	TRANSIENT ANALYSIS
		Execute	EXECUTE
6	System Control	None	*TRACE Note: *TRACE Produces the numbers (or names) of program subroutines as they are entered. It is used for troubleshooting.
		None	*BCD
		None	*EBCDIC
7	Comments	Comment	C any comment necessary to
		Comment	C understand the problem
8	End of Job Instructions	End	END
		None	/*

Table 5-3 (cont.) ECAP/360 TRANSIENT ANALYSIS Input Cards

```
          1   1
/1     7   2   6
 //EXAMPLE JOB (06W1234567Q001,2,2),URSTUCK,MSGLEVEL=1
 //JOBLIB DD DSNAME=SYS1.USERLIB,DISP=OLD
 //STEP1 EXEC PGM=ECAP
 //FT06F001 DD SYSOUT=A,DCB=RECFM=UA
 //FT05F001 DD *
 C
 C      FIRST PROBLEM IN BATCH
        TRANSIENT ANALYSIS

            data

        EXECUTE
 C
 C      SECOND PROBLEM IN BATCH
        TRANSIENT ANALYSIS

            data

        EXECUTE
        TRANSIENT ANALYSIS

            data

        EXECUTE
        TRANSIENT ANALYSIS

            data

        EXECUTE
 C
 C      LAST PROBLEM IN BATCH
        TRANSIENT ANALYSIS

            data

        EXECUTE
        END
 /*
```

Figure 5-7. Example Problem Deck for Solving Transient Problems in a Batch.

the card classification number of each card for quick reference to Table 5-3.

It was mentioned earlier that ECAP transient analysis could approximate the action of an 'ideal' transformer. An 'ideal' transformer is a device that relates its primary and secondary voltages and currents by the equations:

$$V_s = K V_p$$

$$I_s = -I_p / K$$

where:

K is a positive real constant that expresses the 'turns-ratio' of the device,

V_p is the primary voltage,

V_s is the secondary voltage,

I_p is the primary current,

I_s is the secondary current.

If R is a small resistance with respect to other impedances in the circuit, then the equivalent circuit shown in Figure 5-8 will reasonably approximate the 'ideal' transformer. The sources are associated with the branches of R and K^2R. A derivation for an equivalent circuit is shown in the ECAP/1620 USER'S MANUAL, but includes two extra branches not necessary in ECAP/360.

It was mentioned that an ECAP program also exists for the IBM 1620. Since ECAP/360 is a greatly expanded version of ECAP/1620, persons specifically interested in using ECAP/1620 should become familiar with both the ECAP/1620 USER'S MANUAL and the ECAP/1620 SYSTEM MANUAL. An object deck for ECAP/1620,

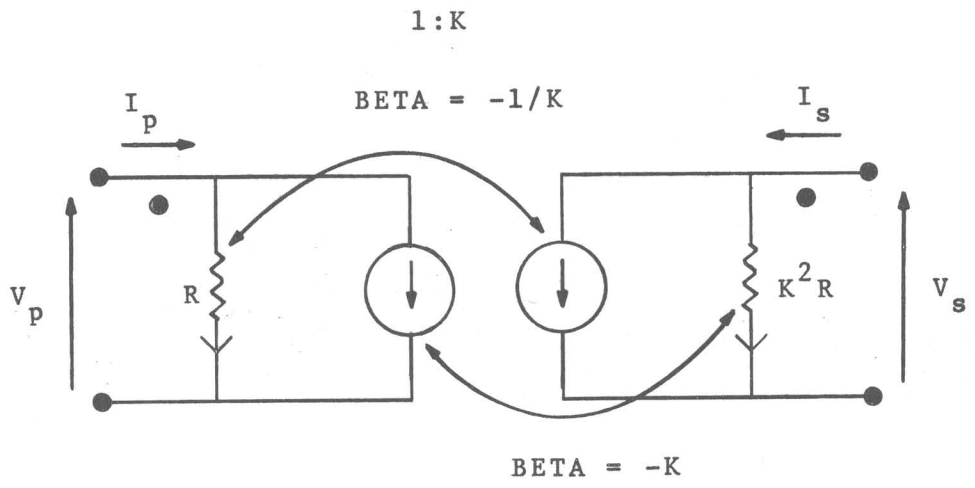


Figure 5-8. Equivalent Circuit for an 'Ideal' Transformer

along with all the ECAP/1620 manuals, are available at the operations office of the KSU Industrial Engineering Department's IBM 1620.

SUMMARY

For DC and AC Analysis calculations, the methods used by ECAP/360 are not too different from those an individual might normally use for pencil and paper solutions. Once an ECAP program user realizes this, the computer becomes a servant rather than a wizard. Even the partial derivative and standard deviation calculations discussed in chapters one and three are not as awesome as they first appear. Once familiar with DC and AC Analysis, the program user has a tendency to forget that solutions for Transient Analysis problems are approximations. The ease with which a non-sinusoidal waveform can be specified 'covers up' the fact that the piecewise linear approximation of the waveform may be fairly poor in some cases. An IBM Technical Newsletter that discusses this waveform error (and some other errors in ECAP/1620 and ECAP/360) is included in this report as an appendix since the newsletter is not now otherwise available to most ECAP program users. A copy of this newsletter was received in personal correspondence with the author of ECAP/360, Mr. Gerald R. Hogsett. The copy received from Mr. Hogsett included informal notes in the margin which indicate whether the specific paragraphs apply to ECAP/360 or not. The newsletter is reproduced as the appendix 'as received', except for the page numbers which have been altered to conform with the remainder of this report. The last paragraph in the newsletter suggests that switching times, 2ERROR, and 3ERROR can interact to invalidate all calculations after a switch

actuation. In correspondence with Mr. Hogsett, he suggested: "I would try to steer clear of 3ERROR. It is not reliable, and is not implemented the same on 1620 and 360."

It is to be expected that the problem with 3ERROR will be corrected, and the ECAP/360 program available at the KSU Computing Center will be updated when the corrections are available. At present, the KSU Computing Center has updated the program to include the set of corrections dated October 27, 1967. These corrections are all that were available as of April 23, 1968. Both DC and AC Analysis are fully implemented; but because of the difficulties with Transient Analysis, it should be reserved for faculty use.

REFERENCES

1. 1620 Electronic Circuit Analysis Program [ECAP] [1620-EE-02X] User's Manual. H20-0170-1. White Plains: International Business Machines Corporation, 1965.

2. IBM 1620 Program Library, Electronics Circuit Analysis Program [ECAP] System Manual, 1620-EE-02X. International Business Machines Corporation.

3. 1620 Electronic Circuit Analysis Program [ECAP] [1620-EE-02X] Application Description. H20-0147-0. White Plains: International Business Machines Corporation.

4. 1620 Electronic Circuit Analysis Programs [ECAP][1620-EE-02X] Operator's Manual. H20-0171-0. White Plains: International Business Machines Corporation.

5. S/360 General Program Library, ECAP/360-E, Electronic Circuit Analysis Program. 360D-16.4.001. International Business Machines Corporation.

6. Falk, Howard. "Computer Programs for Circuit Design." Electro-Technology, March 1964.

7. Branin, Frank. "Analyzing Circuits by the Numbers." Electronics, January 9, 1967.

8. Karni, Shlomo. Network Theory: Analysis and Synthesis. Boston: Allyn and Bacon, 1966, 64-77.

9. Frazer, R. A., W. J. Duncan, and A. R. Collar. Elementary Matrices and some Applications to Dynamics and Differential Equations. Cambridge: Cambridge University Press, 1963, 43-44.

10. Hogg, Robert V., and Allen T. Craig. Introduction to Mathematical Statistics. Second Edition. New York, Macmillan, 1965, 138-139.

11. Crout, Prescott D. "A Short Method for Evaluating Determinants and Solving Systems of Linear Equations with Real or Complex Coefficients." American Institute of Electrical Engineers, Transactions. 60:1235-1241, 1941.

IBM Technical Newsletter

Re: Form No.	H20-0170-1
This Newsletter No.	N20-1036-0
Date	July 17, 1967
Previous Newsletter Nos.	None

1620 ELECTRONIC CIRCUIT ANALYSIS PROGRAM
(ECAP) (1620-EE-02X)
USER'S MANUAL

This newsletter presents precautions regarding the use of certain features for AC and Transient Analysis. These precautions apply only to 1620 ECAP. (See Chapter 6: AC Analysis and Chapter 7: Transient Analysis in the subject manual.)

File this newsletter at the back of the publication. It will provide a reference to changes, a method of determining that all amendments have been received, and a check that the publication contains the proper pages.

IBM Corporation, Technical Publications Dept., 112 E. Post Road, White Plains, N.Y. 10601

N20-1036-0 (H20-0170-1)

Printed in U.S.A.

AC Analysis (Chapter 6 - ECAP User's Manual - H20-0170-1)

1. In the analysis of circuits containing mutual inductors, the nominal solutions are correct. However, modification solutions are incorrect. This is true for all circuits containing mutual inductances, even if the modification does not affect the mutual inductors. It is suggested that the user perform a series of nominal solutions instead of using the MODIFY feature for these cases. These remarks do not apply to circuits containing inductors which are not mutually coupled. 1620 only

2. Voltage sources in series with inductors give incorrect results. It is suggested that the user place the voltage sources in the capacitive or resistive branches in his circuits. "

3. If a circuit contains both dependent current sources and mutual inductors, certain output quantities may be incorrect depending upon the topology of the circuit. Element current, branch current, and element power will be incorrect for those branches that contain a dependent current source, Tnn, when there is also a mutual inductance, Mnn (that is, when the data card sequence numbers are the same for both). "

Transient Analysis (Chapter 7 - ECAP User's Manual - H20-0170-1)

1. If a user requests an EQUILIBRIUM solution for a circuit which contains a time-dependent voltage source with nonzero slope in a capacitive branch, the solution is incorrect. It is suggested that the user replace the time-dependent source by a fixed voltage source in the capacitive branch for the EQUILIBRIUM solution. This source should have a value equal to the first value of the time-dependent source. 1620 only

2. Solutions at TIME = INITIAL TIME (normally 0.0 seconds) are incorrect if a capacitive branch contains a time-dependent voltage source, and the time-dependent source has a slope different from zero. The user should, if possible, move the source to a resistive or inductive branch which is in series with the capacitive branch. If this is not possible, it is suggested that the source start with a slope of zero for its first interval, and then follow the intended waveform after that (see also 3 below). 1620 only

3. The User's Manual states that for the time-dependent sources linear interpolation is used to find source values within the time interval K Δ t. This is a correct statement for circuits that do not contain capacitors. In the analysis of circuits that contain capacitors but no inductors, time-dependent sources are linearly interpolated, but ECAP assumes a stair-step function between integral time steps. 1620 & 360

Figures 1 and 2 illustrate these cases. In Figure 1, the user specified a factor k of 2. ECAP will interpolate and find a value of 5 volts for the source at 0.5 microseconds and at 2.5 microseconds. The source will be treated as a straight-line function between all time steps. Compare this with the R-C circuit of Figure 2. ECAP will treat the time-dependent source as a stair-step with the value of zero at time zero. Just after time zero and up to and including time 0.5 milliseconds,

the source will be assumed to have a value of 5 volts. Just after 0.5 milliseconds, the source will have a value of 10 volts and continue at this value until 2.0 milliseconds. The heavy line in Figure 2(c) illustrates the effect of the time-dependent source card E1 in Figure 2(b).

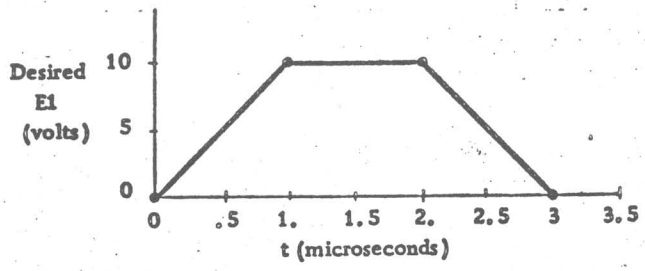


Figure 1 (a). Desired input waveform

```
                TRANSIENT ANALYSIS
B1              N(0,1),R=1000
E1              (2),0,10,10,0
B2              N(1,0),L=1E-3
                TIME STEP=.5E-6
                FINISH TIME=3.5E-6
                PRINT, VOLTAGES
                EXECUTE
```

Figure 1 (b). ECAP input

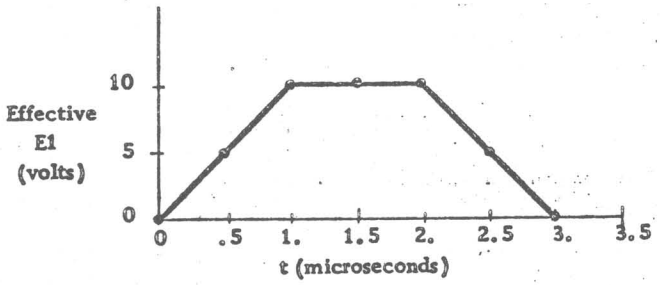


Figure 1 (c). Computational effect

Figure 1. R-L circuit with ECAP time-dependent source

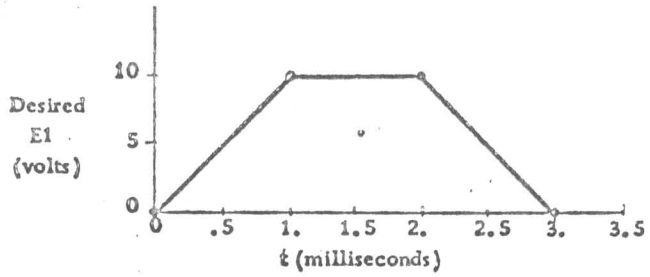


Figure 2 (a). Desired input waveform

```
                TRANSIENT ANALYSIS
B1              N(0,1),R=1000
E1              (2),0,10,10,0
B2              N(1,0),C=1E-6
                TIME STEP=.5E-3
                FINISH TIME=3.5E-3
                PRINT, VOLTAGES
                EXECUTE
```

Figure 2 (b). ECAP input

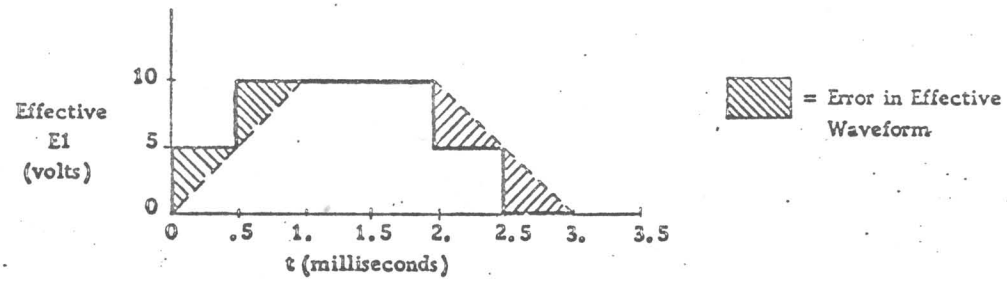
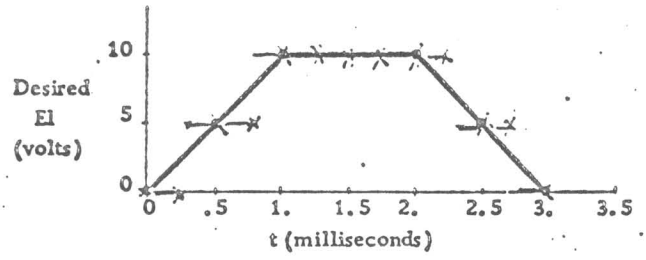


Figure 2 (c). Computational effect

Figure 2. R-C circuit with ECAP time-dependent source

Figure 3 suggests a technique for obtaining the proper calculated response for the circuit of Figure 2. The time-step is set to one-half of the period of the shortest straight-line segment of the time-dependent source, or to one-half of the time wanted between printouts, whichever is shorter. The time-dependent source is now input in the manner shown in Figure 3(b), wherein each straight-line segment of the source is represented by two values. Take, for example, the first segment of the source in Figure 3(a). It is now represented by two segments: one from coordinates (0, 0) to (0.5E-6, 0) and the other from (0.5E-6, 0) to (1E-6, 5). These two segments are treated by ECAP as a stair-step and serve to average out the "Error in Effective Waveform" as can be seen in Figure 3(c). The output interval value should be set to 2 or to any even value so as to eliminate printing of results after each zero slope segment.

A more general technique for reducing the error in the calculated response is that of increasing the factor k for the time-dependent source as much as possible. This would be accomplished by reducing the time-step as much as is practicable. Referring to Figure 2(c), this technique would reduce the size of the "Error in Effective Waveform" by a factor equivalent to k. This technique was employed in the example in Figures 145, 146, 147 in the 1620 ECAP User's Manual (pages 113, 114).



0, 5, 10, 10, 10, 5, 0

Figure 3 (a). Desired input waveform

```
                TRANSIENT ANALYSIS
B1              N(0,1),R=1000
E1              (1),0,0,5,5,10,10,10,10,10,10,5,5,0
B2              N(1,0),C=1E-6
                TIME STEP=.25E-3
                OUTPUT INTERVAL=2
                FINISH TIME=3.5E-3
                PRINT, VOLTAGES
                EXECUTE
```

Figure 3 (b). ECAP input

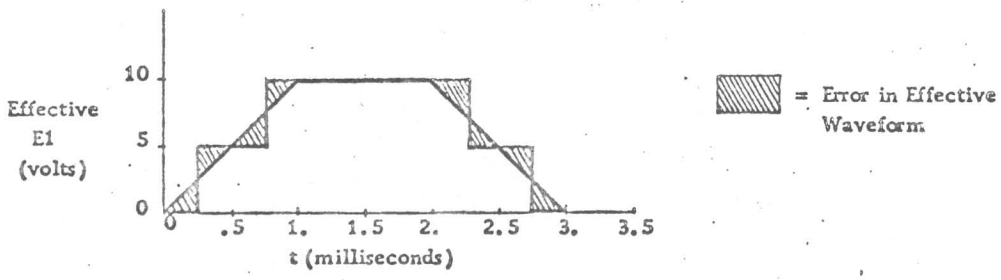


Figure 3 (c). Computational effect

Figure 3. One method of obtaining proper response in R-C circuits

4. In circuits containing time-dependent sources, if switching occurs within 2ERROR of the end of a time-step and this time-step represents a breakpoint in the slope of the time-dependent source, the solutions from this point on are incorrect. This same effect occurs when the user elects to enter a 3ERROR card with a value less than 1.0. It is suggested that, if either of these conditions occur, the user change the time relationship of switching and time-dependent sources. It is indeed rare that the above conditions would occur.

1620
&
360

ACKNOWLEDGEMENTS

Most of the material for this report was gleaned from the publications listed as References, and the author's experience with ECAP/360. However, much assistance was received in interpreting this material and in presenting it in a logical and orderly manner. Mr. Gerald R. Hogsett of the IBM Corporation, the author of ECAP/360, provided the material included as the appendix, a set of corrections to the ECAP/360 source deck, and helpful comments about the operation of the program. Prof. Joseph E. Ward Jr. discovered a most embarrassing error in the earliest manuscript; and the several resulting discussions about circuit conventions and manner of presentation are clearly reflected in the final draft. Dr. Donald H. Lenhert, acting as major advisor, assisted in more ways than could be easily innumerated. Mr. Vijay R. Kumar of the IBM Corporation provided a copy of the ECAP/1620 System Manual when it was most desperately needed. To these individuals, I owe a large debt of gratitude.

VITA

James E. Fraser

Candidate for the Degree of

Master of Science

Report: A GUIDE FOR THE CLASSROOM USE OF IBM'S ECAP/360

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Abilene, Kansas, June 30, 1939, the son of Leslie A. and Mary M. Fraser.

Education: Received Bachelor of Science degree in Electrical Engineering from Kansas State University in February, 1962. Completed requirements for Master of Science in Electrical Engineering in May, 1968.

Professional experience: Entered the United States Army in 1962 with a reserve lieutenancy. Served on active army duty from 1962 to 1964. Employed by Caterpillar Tractor Company as a research engineer from 1965 to 1967. Member of the Institute of Electrical and Electronic Engineers.

A GUIDE FOR THE CLASSROOM USE OF IBM'S ECAP/360

by

JAMES EDWARD FRASER

B. S., Kansas State University, 1962

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1968

ECAP/360 is an IBM distributed, but unsupported, program for IBM System 360 Computers. DC and AC circuits with up to fifty nodes are solved numerically, yielding exact values for circuit voltages, currents, and power. DC circuits can also be solved for node voltage sensitivity coefficients and standard deviations. Transient circuits, those with non-linear elements or excitations, are solved for circuit voltages and currents by repeated piecewise linear approximations. Transsient solutions are received from the computer as circuit conditions at specific times, and can easily be plotted by the program user. The transient solutions are not in a closed, functional form.

The report has two objectives. The first objective is to provide a class instructor with sufficient background information to understand the methods used by ECAP/360 in solving circuits and to remove any aura of mystery or magic about the program. Accordingly, chapters one and two of the report discuss the methods used by ECAP/360 to solve circuits. Included as an appendix to the report is a copy of an IBM Technical Newsletter that covers some solution difficulties not discussed in the material normally furnished with the program. The second objective of this report is to provide short, detailed instruction guides that can be distributed to students. Chapter three is a guide for solving DC circuits with ECAP/360, chapter four is a guide for AC circuits, and chapter five is a guide for Transient circuits. Each of these

three chapters is self-contained. For example; a student can be given chapter four as a classroom handout, and with no additional reference material can be expected to solve AC circuits problems with ECAP/360 at the KSU Computing Center. Included in each of the three chapters are the exact formats for all necessary control cards presently required at the KSU Computing Center and complete example problems. Because of the excellent discussion available in reasonably condensed form in the IBM ECAP/1620 USER'S MANUAL on approximating non-linear devices, chapter five covers only basic transient analysis programming instructions.