

Extended SCEPTRE Volume 2 Mathematical Formulation

David Becker*

October 7, 2018

Revised and edited by
Wolf-Rainer Novender
September 2018 [†]

*GTE Sylvania, Incorporated, Needham Heights, MA 02194

[†]D-64625 Bensheim - Germany

Contents

1	Introduction	2
2	Formulation and Theory	3
2.1	Introduction	3
2.2	General Solution	3
2.3	Transient Solution	6
2.3.1	Matrix Operations	6
2.3.2	Scanning Procedure	9
2.3.3	Semi-automatic Source Derivatives	10
2.3.4	Linearly Dependent Sources	11
2.3.5	Integration Routines	13
2.4	DC Solutions	22
2.4.1	Initial Conditions Solution by Newton-Raphson Method	23
2.4.2	Monte Carlo	28
2.4.3	Sensitivity	29
2.4.4	Worst Case	30
2.4.5	Optimization	32
2.5	AC Solutions	34
2.5.1	Independent Source Only	35
2.5.2	Circuits Containing Linearly Dependent Sources	37
2.5.3	Circuits requiring Time Derivatives of Independent Sources	40
2.5.4	Circuits Containing Linearly Dependent Sources and Requiring Time Derivatives of Independent Sources	41
3	System Operation	42
3.1	Introduction	42
3.2	Program Generator	42
3.2.1	Circuit Description Processor	42
3.2.2	Model Editor	43
3.2.3	Circuit Equation Generator	43
3.2.4	Data Generator and Rerun Processor	43
3.2.5	Continue Processor	43
3.2.6	Re-Output Processor	44
3.3	Solution Executive	44
4	Sparse Matrix Techniques	52
5	Assembler Language Input/Output (I/O)	54
A	Implicit Integration Implementation	56
A.1	Introduction	56
A.2	Manual Example 1	57
A.3	A Practical Illustration	57

A.4	Eigenvalue spread	59
A.5	Jacobian Construction	59
A.6	Preset Step Size Controls	62
B	B Matrix Derivation	63
C	Diode Representation in the Initial-Conditions Program	64
D	Basic Newton-Raphson Method	65
E	Equivalence of Equations (2.79) and (2.80)	66
F	Adjoint Network for Sensitivity	67

List of Figures

2.1	Composite B Matrix in SCEPTRE	4
2.2	Transient-Solution B Matrix in SCEPTRE	6
2.3	B Matrix from Hypothetical Network	10
2.4	Low-Frequency h-Parameter Equivalent Circuit	11
2.5	B Matrix with Linearly Dependent Sources	11
2.6	Root Locus of TRAP Characteristic Equation	16
2.7	Capacitor Cut Set Circuit	26
2.8	Inductor Loop Circuit	27
2.9	A Current-Source Capacitor Cut Set	27
2.10	Two-Dimensional Example of Worst-Case Limits	31
2.11	Change in ΔF Due to Change in Independent Variable	34
3.1	System Flow Diagram	45
3.2	Program Generator (EXEC1) Flow Diagram	46
3.3	Circuit Description Processor (CDPRS) Flow Diagram	47
3.4	Model Editor (MDEDIT) Flow Diagram	48
3.5	Circuit Equation Generator (GENER8) Flow Diagram	49
3.6	Rerun Processor and Data Generator (RRPRO) Flow Diagram	50
3.7	Circuit Solution Executive (EXEC2) Flow Diagram	51
A.1	A Practical Illustration	58
A.2	The General Jacobian	59
F.1	Relation between Network and Adjoint Network (see explanation in table F.1)	67

List of Tables

2.2	Derived Matrices and Vectors from Classification in table 2.1	4
2.1	Element Classifications in SCEPTRE	5
2.3	Comparison of r/p of Different Integration Methods	16
2.4	General B Matrix from L Tree	23
2.5	Sensitivity Computations	29
5.1	Timing Tests of the Setup Phase using FORTRAN I/O and Assembler Language I/O	55
A.1	Comparison of Explicit and Implicit Integration on Example 1	57
A.2	Explicit-Implicit Comparison at Selected Eigenvalue Spreads	59
F.1	Relation between Network and Adjoint Network	68
F.2	Adjoint Network Applied Source	69

This manual is almost an identical copy of [1]. The references to the IBM 7090/94 and S/360 computers have been left untouched for historical reasons. This applies also to expressions like 'cards', 'tapes', 'punch'. The machine dependent chapters (7090/94, S/360, and CDC6600 System Information) have been omitted, references to them within this manual will look like ??.

The figures 3.1 to 3.7 have been directly copied from [1].

1 Introduction

The SCEPTRE Circuit Analysis Program has been developed and improved over a period of years on a number of contracts led by the Air Force Weapons Laboratory. The users' information for the 1973 version of this program is contained in two volumes, of which this is the second .

Volume I [2] contains basic instructions for use of the program. Volume II [1] is intended to give a fuller understanding of the SCEPTRE internal formulation so that the more sophisticated user may take full advantage of the program's capabilities and flexibility. This volume supersedes AFWL-TR-72-77 for use of SCEPTRE on S/360 equipment, and retains the information from AFWL-TR-72-77 pertinent to 7090/94 equipment.

The 1973 version of SCEPTRE, implemented for S/360 use, contains six additions not available on previous versions. There are four new DC options, an AC analysis capability, and a convolution capability. This volume covers the mathematical formulations for all of the previous SCEPTRE features, plus five of the six new ones. The convolution feature is covered entirely in Volume I.

2 Formulation and Theory

2.1 Introduction

Any automatic transient analysis program is designed to relieve the user of the necessity of writing and programing the differential and algebraic equations that describe networks. PREDICT and other programs already perform this basic task, but the degree of flexibility permitted the user varies widely among programs. SCEPTRE, written as a successor to PREDICT, incorporates many improvements.

This section presents the formulation and theory that serve as the basis for SCEPTRE. Therefore, the discussion is mathematically oriented. Those features of SCEPTRE that are not mathematical are not included here. For example the extremely useful features of Rerun and Model Storage are not described.

2.2 General Solution

SCEPTRE consists of three separate formulations that combine to produce the general solutions of a given network. One is referred to as the DC program. It has five options, discussed in subsection 2.4. Each will determine the network voltages that prevail before any time-varying forcing function is applied. This program does not treat time as an independent variable; instead it holds time constant, and iterates on selected voltages. The output of a DC program may be obtained independently, or it may be automatically used as the starting point for the transient or AC program. Thus, the output of the DC program effectively supplies the initial conditions for the system of differential equations that are solved by the transient or AC program.

The second program is called the transient program. This program uses time as an independent variable and solves systems of differential equations as functions of time. The output of this program represents the transient response of a given network. As implied above, the transient program may be used in conjunction with the DC program, or it may be used by itself if the initial conditions of the network are known.

The third program is called the AC program. This program uses frequency as an independent variable and solves algebraic equations as a function of frequency. The output of this program represents the frequency response of a given network. The AC program may also be used in conjunction with the DC program or by itself. It is discussed in Section 2.5.

The general solution procedure described here concerns the definition of the terms, matrices and procedures that are common to all programs. Other parts of this volume will provide the detailed explanations and derivations.

The first step in either program is the construction of a tree¹ [10] according to prescribed rules, which differ for the three programs. This permits formation of a B matrix that effectively expresses link voltages in terms of tree branch voltages and tree branch currents in terms of link currents. Figure 2.1 shows a composite B matrix that contains all possible element classifications and submatrices. This matrix is derived in appendix B. The element classifications are given in table 2.1.

Figure 2.1: Composite B Matrix in SCEPTRE

	Tree Branches				
class	4	5	6	7	Y
1	B_{14}	B_{15}	B_{16}	B_{17}	B_{1Y}
2	B_{24}	B_{25}	B_{26}	B_{27}	B_{2Y}
3	B_{34}	B_{35}	B_{36}	B_{37}	B_{3Y}
8	B_{84}	B_{85}	B_{86}	B_{87}	B_{8Y}
9	B_{94}	B_{95}	B_{96}	B_{97}	B_{9Y}
0	B_{04}	B_{05}	B_{06}	B_{07}	B_{0Y}
X	B_{X4}	B_{X5}	B_{X6}	B_{X7}	B_{XY}

The matrices and vectors shown in table 2.2 may also be defined as a result of the element classification in table 2.1:

Table 2.2: Derived Matrices and Vectors from Classification in table 2.1

R_{22}	a diagonal matrix composed only of resistor links
G_{22}	$= R_{22}^{-1}$
R_{55}	a diagonal matrix composed only of resistor tree branches
G_{55}	$= R_{55}^{-1}$
C_{11}	a diagonal matrix composed only of capacitor links
S_{11}	$= C_{11}^{-1}$
C_{44}	a diagonal matrix composed only of capacitor tree branches
S_{44}	$= C_{44}^{-1}$
L_{33}	A matrix composed only of inductor links and the mutual inductance between inductor links
L_{36}	a matrix composed only of the mutual inductance between inductor links and inductor branches
L_{66}	a matrix composed only of inductor tree branches and the mutual inductance between inductor tree branches
<i>continued on next page ...</i>	

¹A tree is defined as any connected network subgraph that contains all nodes of the network but no complete loops. All circuit elements that are members of the tree are termed tree branches. All circuit elements excluded from the tree are termed links. A "C" tree is defined in this report as one in which tree members are chosen in the preference order E, C, R and L. All current sources (J) must be excluded from the "C" tree. Therefore, these sources are links. A cut set is defined as that group of elements that would isolate two groups of nodes when removed from a network.

Table 2.1: Element Classifications in SCEPTRE

Class	Element
1	Capacitor links
2	Resistor links
3	Inductor links
4	Capacitor tree branches
5	Resistor tree branches
6	Inductor tree branches
7	Voltage sources
8	Independent current sources
9	Primary current sources (dependent on voltage across terminals). This class will appear only in the derivation of the initial conditions and AC programs.
0	Secondary current sources (dependent on other current sources). This class will appear only in the derivation of the initial conditions program.
Y	Voltage sources that are dependent on resistor voltages. This class will appear only in the derivation of the transient and AC programs.
X	Current sources that are dependent on resistor currents. This class will appear only in the derivation of the transient and AC programs.
D	Voltage sources that are generated in adjoint calculations due to the presence of secondary currentsources in the network.

...continued from previous page

G_{99}	a diagonal matrix composed only of the voltage derivatives of primary current sources
I_1, V_1	vectors composed only of the currents or voltages associated with capacitor links
I_2, V_2	vectors composed only of the currents or voltages associated with resistor links
I_3, V_3	vectors composed only of the currents or voltages associated with inductor links
I_4, V_4	vectors composed only of the currents or voltages associated with capacitor tree branches
I_5, V_5	vectors composed only of the currents or voltages associated with resistor tree branches
I_6, V_6	vectors composed only of the currents or voltages associated with inductor tree branches

continued on next page ...

...continued from previous page

J_8, V_8	vectors composed only of the currents or voltages associated with independent current sources
J_9, V_9	vectors composed only of the currents or voltages associated with primary current sources
J_0, V_0	vectors composed only of the currents or voltages associated with secondary current sources
E_7	a vector composed only of voltage sources.

2.3 Transient Solution

2.3.1 Matrix Operations

The state variables of any system can be defined as the minimum set of quantities that will suffice to determine all other quantities in the system at any instant. It can be shown that the knowledge of the set of all capacitor tree branch voltage (V_4) and inductor tree link currents (I_3) is sufficient to determine all other element currents and voltages and, furthermore, that this selection of state variables will allow network formulation in terms of first-order differential equations. The starting point of the derivation then is that quantities V_4 and I_3 are known and the list of unknowns is made up of $V_1, V_2, V_3, V_5, V_6, I_1, I_2, I_4, I_5, I_6, I_7, I_8$. In addition, the derivatives of the state variables, V_4 and I_3 , must be obtained in preparation for the numerical integration routine which produces the updated state variables that are valid at the next time increment.

Some of the equations needed to solve the unknown quantities may be obtained from the transient solution B matrix (figure 2.2). The B matrix itself arises from a "C" tree, which is formed by an E, C, R, L preference order. Note that the B matrix differs from the composite matrix of figure 2.1 in that some submatrices are zero valued² and that no distinction is made between types of sources. Since link voltages can be written in terms of tree branch voltages directly from the B matrix, the

Figure 2.2: Transient-Solution B Matrix in SCEPTRE

		Tree Branches				
		class	4	5	6	7
Links	1	B_{14}	0	0	B_{17}	
	2	B_{24}	B_{25}	0	B_{27}	
	3	B_{34}	B_{35}	B_{36}	B_{37}	
	8	B_{84}	B_{85}	B_{86}	B_{87}	

²For example, B_{15} must be zero since the "C" tree preference prohibits the possibility of a capacitor link closing a loop that contains a resistor tree branch.

following equations may be written in matrix form:

$$V_1 = -B_{14}V_4 - B_{17}E_7 \quad (2.1)$$

$$V_2 = -B_{24}V_4 - B_{25}V_5 - B_{27}E_7 \quad (2.2)$$

$$V_3 = -B_{34}V_4 - B_{35}V_5 - B_{36}V_6 - B_{37}E_7 \quad (2.3)$$

$$V_8 = -B_{84}V_4 - B_{85}V_5 - B_{86}V_6 - B_{87}E_7 . \quad (2.4)$$

Since tree branch currents can be written in term of link currents, there arises:

$$I_4 = B_{14}^T I_1 + B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8 \quad (2.5)$$

$$I_5 = B_{25}^T I_2 + B_{35}^T I_3 + B_{85}^T J_8 \quad (2.6)$$

$$I_6 = B_{36}^T I_3 + B_{86}^T J_8 \quad (2.7)$$

$$I_7 = B_{17}^T I_1 + B_{27}^T I_2 + B_{37}^T I_3 + B_{87}^T J_8 \quad (2.8)$$

where the superscript T is used to indicate the transpose of a matrix.

Two additional equations may be obtained from differentiation Equations (2.1) and (2.7) yielding:

$$\dot{V}_1 = -B_{14}\dot{V}_4 - B_{17}\dot{E}_7 \quad (2.9)$$

$$\dot{I}_6 = B_{36}^T \dot{I}_3 + B_{86}^T \dot{J}_8 . \quad (2.10)$$

Note that source derivations $\dot{E}_7$ and $\dot{J}_8$ have been introduced in the last two equations³. These equations, together with a few fundamental relations, will be used to derive all of the network currents and voltages in terms of known quantities.

Solution of Resistive Quantities

The resistive quantities of interest are I_2, I_5, V_2, V_5 . The fundamental voltage current relations for resistors permit:

$$V_2 = R_{22}I_2 \quad (2.11)$$

$$V_5 = R_{55}I_5 . \quad (2.12)$$

I_2 may be explicitly solved for by manipulation of equations (2.2), (2.6), (2.11) and (2.12) to get

$$I_2 = M_R^{-1} \{-B_{24}V_4 - B_{27}E_7 - B_{25}R_{55} [B_{35}^T I_3 + B_{85}^T J_8]\} \quad (2.13)$$

where

$$M_R = R_{22} + B_{25}R_{55}B_{25}^T . \quad (2.14)$$

The significance of equation (2.13) is that the vector of resistor link currents has been solved in terms of all known quantities, since the right side of the equation is composed entirely of state variables V_4 and I_3 , known sources E_7 and J_8 and known incidence submatrices. Once I_2 is known, the vectors I_5, V_2, V_5 can be determined from Equations (2.6), (2.11), and (2.12), respectively.

An alternate approach may sometimes be preferable. Equation (2.2), (2.6), (2.11), and (2.12) may also be manipulated to obtain

$$V_5 = M_G^{-1} \{-B_{25}^T G_{22} [B_{24}V_4 + B_{27}E_7] + B_{35}^T I_3 + B_{85}^T J_8\} \quad (2.15)$$

³See subsection 2.3.3 for a discussion on source derivatives.

where

$$M_G = G_{55} + B_{25}^T G_{22} B_{25} . \quad (2.16)$$

Equation (2.15) gives the vector of resistor tree branch voltages in term of quantities that are all known. Then, V_2, I_2, I_5 can be solved by Equation (2.2), yielding

$$I_2 = G_{22} V_2 \quad (2.17)$$

$$I_5 = G_{55} V_5 . \quad (2.18)$$

The two approaches differ in the size of the matrix to be inverted. The first approach requires the inversion of a matrix M_R containing the number of rows and columns equal to the number of class-2 elements in the network. The second approach requires the inversion of a matrix M_G containing the number of rows and columns equal to the number of class-5 elements in the network. Networks containing resistors that are all constant require only one matrix inversion. There is no practical difference between the two approaches. If, however, the network contains at least one variable resistor, a matrix must be inverted at each solution time increment. Hundreds, or even thousands, of matrix inversions are necessary and the size of the matrix becomes of significant importance. SCEPTRE will automatically determine which of the two approaches should be taken for each individual network.

Solution of Capacitor Quantities

The capacitor quantities that must be solved at each time step are I_1, I_4, V_1, V_4 . Vector V_4 itself will have been updated by the integration routine and, hence, will be known. The fundamental relationships for capacitors permit

$$\dot{V}_4 = S_{44} I_4 \quad (2.19)$$

$$\dot{V}_1 = S_{11} I_1 . \quad (2.20)$$

Equations (2.5), (2.9), (2.19) and 2.2 may be combined to obtain

$$I_1 = M_S^{-1} \{ -B_{14} S_{44} [B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8] - B_{17} \dot{E}_7 \} \quad (2.21)$$

where

$$M_S = S_{11} + B_{14} S_{44} B_{14}^T . \quad (2.22)$$

At this point, vector I_1 has been isolated in terms of all known quantities. There remains to obtain I_4, V_1, V_4 from equations (2.5), (2.1), and (2.19), respectively.

Solution of Inductive Quantities

The inductive quantities that must be solved at each time step are I_3, V_3, V_6, I_6 . Vector I_3 will have been updated at the start of each time step and will be known. The fundamental relations for inductors permit

$$V_3 = L_{33} \dot{I}_3 + L_{36} \dot{I}_6 \quad (2.23)$$

$$V_6 = L_{63} \dot{I}_3 + L_{66} \dot{I}_6 . \quad (2.24)$$

Equations (2.3), (2.10), (2.23), and (2.24) may be combined to obtain

$$\dot{I}_3 = M_L^{-1} \{ -B_{34}V_4 - B_{35}V_5 - B_{37}E_7 - [B_{36}L_{66}B_{86}^T + L_{36}B_{86}^T] \dot{J}_8 \} \quad (2.25)$$

where

$$M_L = L_{33} + L_{36}B_{36}^T + B_{36}L_{63} + B_{36}L_{66}B_{36}^T. \quad (2.26)$$

Now $\dot{I}_3$ is written in terms of known quantities. Following this, V_3, V_6, I_6 may be obtained from Equations (2.10) and (2.23), (2.10) and (2.24), and (2.7), respectively.

Solution of Voltage Source Currents and Current Source Voltages

A complete list of possible outputs of a network would include the current through voltage sources and the voltage across current sources. These can be obtained directly from Equations (2.8) and (2.4), respectively, since the right sides of these equations are known at this stage of the computational sequence. These steps complete the formal derivation of all network currents and voltages.

2.3.2 Scanning Procedure

Ideal Operation (Submatrices B_{14}, B_{25} , and $B_{36} = 0$)

The series of matrix operations described in subsection 2.3.1 could very well be programmed as they are to produce the solution of the general transient analysis problem. All of the matrix multiplication, addition, etc. could be performed at each time step to generate the necessary currents and voltages. However, a very significant improvement in computer running time could be achieved by a more efficient utilization of the information contained in the B matrix.

Various network voltages and currents could be read or "scanned" directly from the rows and columns of the B matrix respectively⁴. This can be put more precisely by

$$V_L = -B V_{TB} \quad (2.27)$$

and

$$I_{TB} = B^T I_L \quad (2.28)$$

where I_{TB}, I_L, V_{TB} and V_L represent the vectors of tree branch currents, link currents, tree branch voltages, and link voltages, respectively. If the vectors and the B matrix are partitioned according to the form of figure 2.2, Equations (2.27) and (2.28) lead to the first six equations of subsection 2.3.1. Quantities V_2 and I_5 are explicitly written in terms of known quantities of Equations (2.2) and (2.6) if submatrix $B_{25} = 0$. Once the resistive quantities are determined, equation (2.5) presents I_4 in terms of known quantities if submatrix $B_{14} = 0$. In the same manner, equation (2.3) presents V_3 in terms of known quantities if submatrix $B_{36} = 0$. Clearly, then, the condition " B_{14}, B_{25} , and $B_{36} = 0$ " permits solution of $V_2, I_5, I_4, V_3, I_6, V_1$ directly by scanning and without any matrix manipulations. Quantities $I_2, V_5, I_1, V_6, V_4, I_3$ can then be determined from the operations given or implied in the non zero portions of Equations (2.11), (2.12), (2.21), (2.24), (2.19), and (2.25), respectively. Once all of these quantities are explicitly obtained in terms of known quantities, the equations are stored, compiled, and executed at each time increment without recourse to repeated matrix manipulation.

⁴AFWL-TR-65-101, Volume I

Operation When $B_{25} \neq 0$

In most large networks, submatrix B_{25} does not equal zero. When this happens, quantities V_2 and I_5 cannot be "scanned" out and either Equation (2.13) or (2.15) must be solved. Both of these equations require the inversion of a matrix; thus, some matrix manipulation must be done. To illustrate the procedure, assume that some hypothetical network has given rise to the B matrix shown in figure 2.3. The network is fairly typical in that $B_{14}, B_{36} = 0$, but $B_{25} \neq 0$. The nature

Figure 2.3: B Matrix from Hypothetical Network

		Class 4		Class 5				Class 7
		C_1	C_2	R_A	R_B	R_C	R_D	E_1
Class 2	R_1	1	0	1	0	0	0	-1
	R_2	0	1	0	1	1	0	0
	R_3	0	1	0	0	0	0	-1
	R_4	0	0	1	0	0	0	0
Class 3	L_1	0	0	0	0	0	1	0
	L_2	0	0	1	0	0	0	0

of submatrix B_{25} (outlined in figure 2.3) is such that resistors R_3 and R_D contribute no rows or columns, and the scanning process immediately yields

$$V_{R3} = E_1 - V_{C2}$$

$$I_{RD} = I_{L1}$$

from which follow

$$I_{R3} = (E_1 - V_{C2})/R_3$$

$$V_{RD} = I_{L1} R_D .$$

These quantities will be updated at each time step without matrix manipulation. The rest of the resistive quantities in the network may be solved by the matrix manipulation implied in Equations (2.13) and (2.14). Once the resistive quantities are determined, the remaining network quantities may be scanned out and stored. The only repeated matrix manipulation required will be for the three resistor link currents (I_{R1}, I_{R2}, I_{R4}) that cannot be scanned because $B_{25} \neq 0$.

2.3.3 Semi-automatic Source Derivatives

The need for source time derivatives in transient analyses is established in Equations (2.9) and (2.10) where E_7 and $\dot{J}_8$ are required. In situations where non-zero source time derivatives are needed, the user must supply them. These situations are subsequently described.

Voltage Source Is Variable and $B_{17} \neq 0$

If $B_{17} \neq 0$ and the voltage source is constant, SCEPTRE will automatically supply a zero derivative. In the case where a non-zero derivative is required and the user fails to supply it, the run will be

terminated with a diagnostic message. The situation is best recognized by the presence of any circuit loop composed solely of capacitors and at least one variable voltage source.

Current Source Is Variable and $B_{86} \neq 0$

If $B_{86} \neq 0$ and the current source is constant, SCEPTRE will automatically supply a zero derivative. In the case where a non-zero derivative is required and the user fails to supply it, the run will be terminated with a diagnostic message. The situation is best recognized by the presence of any circuit cut set composed solely of inductors and at least one variable current source.

2.3.4 Linearly Dependent Sources

The ability of SCEPTRE to correctly process linearly dependent sources permits the user to define current and voltage sources that are linear functions of resistor currents and voltages. This feature is expected to be most useful for, but not limited to, applications involving families of small-signal transistor equivalent circuits such as shown in figure 2.4.

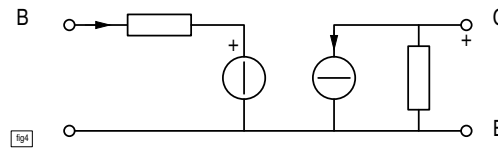


Figure 2.4: Low-Frequency h-Parameter Equivalent Circuit

These linearly dependent sources require special treatment because their magnitudes are direct functions of quantities that are not state variables. Unless these sources are specifically processed, they will be updated at the n th time step according to the values of their independent variables at the $(n - 1)$ time step (as they would be in PREDICT, for example). This results in a "computational delay", which can lead to large errors throughout the entire network.

Linearly dependent sources are provided for in the general B matrix by the Y and X classification. When these sources exist, SCEPTRE will set up the B matrix as shown in figure 2.5.

Figure 2.5: B Matrix with Linearly Dependent Sources

class	4	5	6	7	Y
1	B_{14}	0	0	B_{17}	B_{1Y}
2	B_{24}	B_{25}	0	B_{27}	B_{2Y}
3	B_{34}	B_{35}	B_{36}	B_{37}	B_{3Y}
8	B_{84}	B_{85}	B_{86}	B_{87}	B_{8Y}
X	B_{X4}	B_{X5}	B_{X6}	B_{X7}	B_{XY}

Under these circumstances, Equations (2.2) and (2.6) can be extended to:

$$V_2 = -B_{24}V_4 - B_{25}V_5 - B_{27}E_7 - B_{2Y}E_Y \quad (2.29)$$

$$I_5 = B_{25}^T I_2 + B_{35}^T I_3 + B_{85}^T J_8 + B_{X5}^T J_X. \quad (2.30)$$

Since any resistor-voltage-dependent voltage source must depend on a resistor tree branch voltage or a resistor link voltage, there arises

$$E_Y = k_1 V_2 + k_2 V_5 \quad (2.31)$$

where:

- k_1 is a matrix of constants containing the number of rows equal to the number of class-Y voltage sources and the number of columns equal to the number of non-scannable class-2 elements.
- k_2 is a matrix of constants containing the number of rows equal to the number of class-Y voltage sources and the number of columns equal to the number of non-scannable Class-5 elements.
- E_Y is a vector composed of class-Y voltage sources.

Also, since any resistor-current-dependent current source must depend on a resistor tree branch current or a resistor link current, there arises

$$J_X = k_3 I_2 + k_4 I_5 \quad (2.32)$$

where

- k_3 is a matrix of constants containing the number of rows equal to the number of class-X current sources and the number of columns equal to the number of non-scannable class-2 elements
- k_4 is a matrix of constants containing the number of rows equal to the number of class-X current sources and the number of columns equal to the number of non-scannable class-5 elements
- J_X is a vector composed of class-X current sources.

If Equations (2.31) and (2.32), together with $V_2 = R_{22}I_2$ and $I_5 = G_{55}V_5$ are substituted into Equations (2.29) and (2.30)

$$R_{22}I_2 + B_{25}V_5 + B_{2Y}[k_1 R_{22}I_2 + k_2 V_5] = -B_{24}V_4 - B_{27}E_7 \quad (2.33)$$

$$G_{55}V_5 - B_{25}^T I_2 - B_{X5}^T [k_3 I_2 + k_4 G_{55}V_5] = B_{35}^T I_3 + B_{85}^T J_8 \quad (2.34)$$

The last two equations may be consolidated as

$$\begin{bmatrix} R_{22} + B_{2Y}k_1 R_{22} & B_{25} + B_{2Y}k_2 \\ -B_{25}^T - B_{X5}^T k_3 & G_{55} - B_{X5}^T k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = \begin{bmatrix} -B_{24}V_4 - B_{27}E_7 \\ B_{35}^T I_3 + B_{85}^T J_8 \end{bmatrix}.$$

If the large matrix on the left side is called M_{RG} , then

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = M_{RG}^{-1} \begin{bmatrix} -B_{24}V_4 - B_{27}E_7 \\ B_{35}^T I_3 + B_{85}^T J_8 \end{bmatrix}$$

and the resistive quantities I_2 and V_5 can be determined without computational delay. Note that since all four "k" matrices are constrained to be constant, the linearly dependent source feature itself will not require any more than one inversion of M_{RG} . If any variable resistors are present in a network, M_{RG} must, of course, be inverted at each time step. In addition, the extra row and column that was added to the B matrix of figure 2.5 by the linearly dependent sources will add terms to

the equations used in the solution of capacitive, inductive and some source quantities. Specifically, Equations (2.5), (2.3), (2.4) and (2.8) become

$$I_4 = B_{14}^T I_1 + B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8 + B_{X4}^T J_X \quad (2.35)$$

$$V_3 = -B_{34} V_4 - B_{35} V_5 - B_{36} V_6 - B_{37} E_7 - B_{3Y} E_Y \quad (2.36)$$

$$V_8 = -B_{84} V_4 - B_{85} V_5 - B_{86} V_6 - B_{87} E_7 - B_{8Y} E_Y \quad (2.37)$$

$$I_7 = B_{17}^T I_1 + B_{27}^T I_2 + B_{37}^T I_3 + B_{87}^T J_8 + B_{X7}^T J_X \quad (2.38)$$

and the new relations

$$V_X = -B_{X4} V_4 - B_{X5} V_5 - B_{X7} E_7 - B_{XY} E_Y \quad (2.39)$$

$$I_Y = B_{2Y}^T I_2 + B_{3Y}^T I_3 + B_{8Y}^T J_8 + B_{XY}^T J_X \quad (2.40)$$

now exist. A restriction must be placed on these sources based on the content of section 2.5. The B matrix of figure 2.5 transforms Equations (2.9) and (2.10) into:

$$\dot{V}_1 = -B_{14} \dot{V}_4 - R_{17} \dot{E}_7 - B_{1Y} \dot{E}_Y \quad (2.41)$$

$$\dot{I}_6 = B_{36}^T \dot{I}_3 + B_{86}^T \dot{J}_8 + B_{X6}^T \dot{J}_X. \quad (2.42)$$

Now, additional time derivatives E_Y and J_X are required whenever B_{1Y} or $B_{X6} \neq 0$.

Differentiation of Equations (2.31) and (2.32) would involve quantities $\dot{V}_2, \dot{V}_5, \dot{I}_2, \dot{I}_5$. The formulation contains no provisions for these quantities, and there is no way the user could know them to supply them as input data. SCEPTRE will automatically check for the existence of non-zero B_{1Y} or B_{X6} and terminate the run with a diagnostic message when they occur. The situation can be recognized by the presence of any circuit loop composed solely of capacitors and at least one linearly dependent voltage source, or the presence of any circuit cut set composed solely of inductors and at least one linearly dependent current source.

2.3.5 Integration Routines

Four integration routines are optionally available for use in SCEPTRE. Two of these, RUK and TRAP, were available in PREDICT and have been only slightly modified. The third routine, called XPO, was developed at the IBM Scientific Center in Palo Alto, California, by Dr. R. Warten and Mr. H. Fowler and adapted for use in SCEPTRE. Studies to date indicate that XPO is usually, although not always, faster than the other methods. For that reason, this routine will always be used unless the user explicitly requests otherwise in the RUN CONTROL section of any run. The fourth integration routine, IMPLICIT, is discussed in subsection 2.3.5. This routine has advantages when the spread between the largest and smallest time constants in a network to be analyzed is very large.

RUK Formulas and Variable Step Size Control

The well-known Runge-Kutta fourth-order-accuracy formulas ([11]) for the numerical solution of the differential equation

$$\dot{y} = f(t, y)$$

are given by

$$y(t+h) = y(t) + \frac{1}{6} [k_1 + 2k_2 + 2k_3 + k_4] \quad (2.43)$$

where

$$\begin{aligned} k_1 &= hf[t, y(t)] \\ k_2 &= hf\left[t + \frac{h}{2}, y(t) + \frac{1}{2}k_1\right] \\ k_3 &= hf\left[t + \frac{h}{2}, y(t) + \frac{1}{2}k_2\right] \\ k_4 &= hf[t + h, y(t) + k_3] . \end{aligned}$$

Variable step size control ([12]) is achieved by computing

$$k_5 = hf[t + h, y(t + h)]$$

and

$$E = k_1 - 2k_3 - 2k_4 + 3k_5 . \quad (2.44)$$

The above formulas easily generalize to systems of differential equations, in which case y and E become

$$y(t) = [y_1(t), \dots, y_n(t)]$$

$$E = (e_1, \dots, e_n) .$$

Now let

$$u_1, u_2, l_1, l_2 \geq 0$$

Set

$$U_k = u_1 + u_2 |y_k(t + h)|$$

$$L_k = l_1 + l_2 |y_k(t + h)| .$$

- If $|e_k| > 1.5U_k$ for some k , the integration step h is halved, the independent variable is restored from $t + h$ to t , and the values of y and $\dot{y}$ are restored to the values at time t .
- If $|e_k| > 0.75U_k$ for some k , and $|e_k| \leq 1.5U_k$ for all k , the current integration step is accepted, but the stepsize is halved for succeeding steps.
- If $L_k \leq |e_k| \leq 0.75U_k$ for all k , the step size is unaltered.
- If $|e_k| \leq U_k$ for all k , and $|e_k| < L_k$ for at least one k , a doubling indicator is activated . Actual doubling is delayed for seven time steps. Halving always takes precedence over doubling; thus, anytime a halving signal is received, the step size is halved and doubling is delayed for at least seven steps. Similarly, after successful doubling, another seven steps must elapse before the step size can be doubled again.

Recommended choices for u_1, u_2, l_1, l_2 are as follows. If absolute error control is desired, set

$$u_1 = 0.0075 \quad u_2 = 0 \quad l_1 = 0.00005 \quad l_2 = 0 .$$

If relative error control is desired, set

$$u_1 = 0.005 \quad u_2 = 0.005 \quad l_1 = 0.00005 \quad l_2 = 0.00005 .$$

If smaller step sizes are desired than the ones yielded by the above settings, the 0.0075 and 0.00005 settings should be reduced by a factor of 32. This will yield half the previous step sizes.

Modified Trapezoidal Integration (TRAP)

Method Given the differential equation

$$\dot{y}(t) = f(t, y(t)), \quad y(0) = y_0$$

consider the following numerical integration scheme,

$$y_p \left(t + \frac{3}{2}h \right) = y(t) + \frac{h}{2} \dot{y}(t) \quad (2.45)$$

$$\dot{y}_p \left(t + \frac{3}{2}h \right) = f \left[t + \frac{3}{2}h, y_p \left(t + \frac{3}{2}h \right) \right] = f \left[t + \frac{3}{2}h, y(t) + \frac{h}{2} \dot{y}(t) \right] \quad (2.46)$$

$$y_c(t+h) = y(t) + \frac{h}{3} \left[2\dot{y}(t) + \dot{y}_p \left(t + \frac{3}{2}h \right) \right] \quad (2.47)$$

$$\dot{y}_c(t+h) = f[t+h, y_c(t+h)]. \quad (2.48)$$

Step size control is achieved by computing

$$E = \frac{2}{3}h \left| \dot{y}_p \left(t + \frac{3}{2}h \right) - \dot{y}(t) \right|. \quad (2.49)$$

The magnitude of $|E|$ indicates any changes to be made in step size.

Truncation Error Inasmuch as $\ddot{y} = f_t + \dot{y}f_y$ and the true solution y_T is approximated by

$$y_T(t+h) = y(t) + h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t)$$

one obtains,

$$y_T(t+h) = y(t) + h\dot{y}(t) + \frac{h^2}{2}(f_t + \dot{y}f_y). \quad (2.50)$$

On the other hand

$$f \left[t + \frac{3h}{2}, y(t) + \frac{h}{2}\dot{y}(t) \right] \doteq f(t, y(t)) + \frac{3h}{2}f_t + \frac{h}{2}\dot{y}f_y \quad (2.51)$$

whence Equation (2.47) becomes

$$y_c(t+h) \doteq y(t) + \frac{h}{3} \left[2\dot{y}(t) + \dot{y}(t) + \frac{3h}{2}f_t + \frac{h}{2}\dot{y}f_y \right].$$

Subtracting Equation (2.47) from (2.50) yields

$$y_T(t+h) - y_c(t+h) \doteq \left(\frac{h^2}{2} - \frac{h^2}{6} \right) \dot{y}f_y = \frac{h^2}{3} \dot{y}f_y$$

which, in turn, yields the approximate local truncation error.

From Equation (2.46) and (2.51) one gets

$$\dot{y}_p \left(t + \frac{3h}{2} \right) - \dot{y}(t) \doteq \frac{3h}{2}f_t + \frac{h}{2}\dot{y}f_y$$

whence

$$\frac{2h}{3} \left[\dot{y} \left(t + \frac{3h}{2} \right) - \dot{y}(t) \right] \doteq h^2 f_t + \frac{h^2}{3} \dot{y}f_y. \quad (2.52)$$

One notes that if $f_t = 0$, then Equation (2.52) is a good representation of the local truncation error, neglecting higher order terms.

Experience indicates that, for systems of equations of the form $\dot{Y} = A(Y)Y + B$ where A is piecewise constant, the method as well as step-size control is adequate.

Stability Consider the differential equation

$$\dot{y} = -ay, \quad y(0) = y_0, \quad a > 0$$

The numerical integration scheme given by Equation (2.45) through (2.48) yields

$$y_c(nh) = \left(1 - ah + \frac{(ah)^2}{6}\right)^n y(0).$$

This is easily established by induction. Thus, for numerical stability it is necessary and sufficient that

$\left|1 - ah + \frac{(ah)^2}{6}\right| < 1$. This yields $0 < ah < 6$ as shown in figure 2.6. Inasmuch as $|ah| < 6$ is necessary

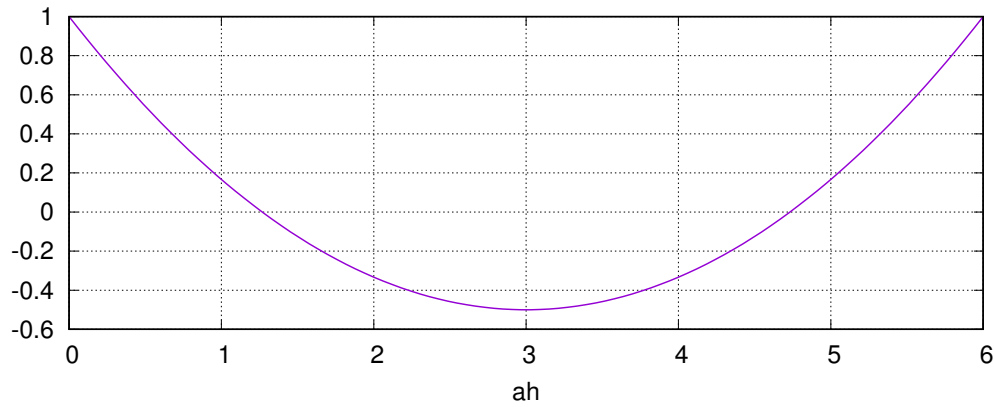


Figure 2.6: Root Locus of TRAP Characteristic Equation

and sufficient for numerical stability, we say that the modified trapezoidal method has a stability radius, r , equal to 6. Moreover, from Equations (2.45) and (2.46) we see that the method requires two derivative evaluations (passes) per integration step. Thus, the pass number p associated with the method is 2.

The ratio r/p is a measure of a method's efficiency in the sense of minimum number of integration steps per given time interval. The higher the r/p , the fewer steps are required for a given a . Table 2.3 shows $r, p, r/p$ for a few representative methods.

Table 2.3: Comparison of r/p of Different Integration Methods

Method	r	p	r/p
Modified TRAP	6	2	3
Euler [13]	2	1	2
Trapezoidal [14]	2	2	1
NIDE (1-Pass) [15]	0.8	1	0.8
Runge-Kutta Fourth-Order	2.78	4	0.7
NIDE (2-Pass)	1	2	0.5
Hamming [16]	0.85	2	0.425

Step-Size Control As mentioned previously, the quantity $E_2 = \frac{2}{3}h |\dot{y}_p(t + \frac{3}{2}h) - \dot{y}(t)|$ is used for step size control.

The method is essentially the same as for RUK. Let u_1, u_2, l_1, l_2 be real numbers ≥ 0 .

1. Compute

$$U = u_1 + u_2 |y_c(t + h)|, \quad L = l_1 + l_2 |y_c(t + h)|.$$

- If $|E_2| > 0.75U$, the step size is reduced.
- If $|E_2| < L$, the step size is increased.
- If $L \leq |E_2| \leq 0.75U$, the step size is not altered.

2. Choose u_1, u_2, l_1, l_2 . Recall that E_2 attempts to represent the local truncation error. For absolute error control set

$$u_1 = 0.01 \quad u_2 = 0 \quad l_1 = 0.0005 \quad l_2 = 0.$$

For relative error control set

$$u_1 = 0.001 \quad u_2 = 0.01 \quad l_1 = 0.00005 \quad l_2 = 0.0005.$$

The above values work well in practice.

Reducing the above settings by a factor of 4 will reduce the step size by a factor of 2.

Exponential Integration (XPO)

Method Given the differential equation

$$\dot{y}(t) = f[t, y(t)], \quad y(0) = y_0$$

this integration routine gives the computed solution

$$y(t + h) = y(t) + \dot{y}_A(t)h + \frac{e^{\lambda h} - 1}{\lambda h} h \dot{y}_p(t) \quad (2.53)$$

except for two cases that will be covered later. In Equation (2.53)

$$\dot{y}_A(t) = \frac{y(t) - y(t - h_0)}{h_0}$$

where $t - h_0$ is the last point computed:

$$\dot{y}_p(t) = \dot{y}(t) - \dot{y}_A(t) \quad (2.54)$$

$$\lambda = \frac{\ddot{y}_p(t)}{\dot{y}_p(t)}. \quad (2.55)$$

Since $\ddot{y}(t)$ cannot be computed explicitly by the program, we take a small Euler integration step and approximate $\ddot{y}(t)$ by $\ddot{y}_\delta(t)$:

$$\begin{aligned} y_\delta(t + \delta) &= y(t) + \delta \dot{y}(t) \\ \dot{y}_\delta(t + \delta) &= f[t + \delta, y_\delta(t + \delta)] \\ \ddot{y}_\delta(t) &= \frac{\dot{y}_\delta(t + \delta) - \dot{y}(t)}{\delta} \end{aligned} \quad (2.56)$$

with $0 < \delta \leq \frac{h}{4}$.

The first exception occurs if $\ddot{y}_\delta(t)$ and $\dot{y}_p(t)$ have the same sign. Then set

$$\dot{y}_A(t) = 0$$

which results in Equations (2.54) and (2.55) becoming

$$\begin{aligned} \dot{y}_p(t) &= \dot{y}(t) \\ \lambda &= \frac{\ddot{y}_\delta(t)}{\dot{y}(t)} . \end{aligned} \quad (2.57)$$

This provides a better approximation to the solution of the equation $\ddot{y} = \lambda y + b$ without decreasing the overall effectiveness of the method.

The second exception occurs if λ , computed either by Equation (2.55) or (2.57), is non-negative. Then the term $\frac{e^{\lambda h} - 1}{\lambda h}$ in Equation (2.53) is replaced by $1 + \frac{\lambda h}{2}$, thus changing Equation (2.53) to

$$y(t + h) = y(t) + \dot{y}_A(t)h + \left(1 + \frac{\lambda h}{2}\right) h \dot{y}_p(t) . \quad (2.58)$$

It can be shown with some effort that Equation (2.58) is equivalent to a trapezoidal type integration method.

Truncation Error Noting that $\frac{e^x - 1}{x}$, when expanded in a Taylor series, becomes

$$\frac{e^x - 1}{x} = 1 + \frac{x}{2} + \frac{x^2}{6} + O(x^3)$$

Equation (2.53) can be written

$$\begin{aligned} y(t + h) &= y(t) + \dot{y}_A(t)h + \dot{y}_p(t)h + \frac{h^2}{2} \lambda \dot{y}_p(t) + \frac{h^3 \lambda^2 \dot{y}_p(t)}{6} + O(h^4) \\ &= y(t) + \dot{y}(t)h + \frac{h^2}{2} \ddot{y}_\delta(t) + \frac{h^3}{6} \lambda \ddot{y}_\delta(t) + O(h^4) \end{aligned} \quad (2.59)$$

Referring to Equation (2.56), note that

$$\ddot{y}_\delta(t) = \ddot{y}(t) + \frac{\delta}{2} \left[\frac{d^3 y}{dt^3} - \ddot{y}(t) \frac{\partial f}{\partial y} \right] + O(\delta^2) . \quad (2.60)$$

Substituting Equation (2.60) into Equation (2.59) yields

$$y(t + h) = y(t) + \dot{y}(t)h + \ddot{y}(t) \frac{h^2}{2} + \frac{h^2}{4} \delta \left[\frac{d^3 y}{dt^3} - \ddot{y}(t) \frac{\partial f}{\partial y} \right] + O(h^3) . \quad (2.61)$$

It is noted that the use of either exception to the principal Equation (2.53) still yields the truncated expression, Equation (2.61).

Since $\delta \leq \frac{h}{4}$, the method is second order exact; i. e. the error is of order h^3 . This characteristic will be used in Equation (2.62) in the next section.

Step-Size Control For $t \leq t + \xi \leq t + h$, let $y_T(t + \xi)$ be the true solution to the differential equation, let $y_c(t + \xi)$ be the computed solution, let $\dot{y}(t + \xi)$ be an expression obtained by differentiation of $y_e(t + \xi)$ with respect to ξ ; i. e.,

$$\dot{y}_e(t + \xi) = \begin{cases} \dot{y}_A(t) + (1 + \lambda\xi)\dot{y}_p(t), & \lambda \geq 0 \\ \dot{y}_A(t) + e\lambda\xi\dot{y}_p(t), & \lambda < 0 \end{cases}$$

and let $\dot{y}_c(t + \xi)$ be obtained by substituting $y_c(t + \xi)$ into the differential equation.

Ideally, step control should be based on the expression

$$E_0 = \int_0^h [\dot{y}_T(t + \xi) - \dot{y}_e(t - \xi)] d\xi$$

however, $\dot{y}_T(t + \xi)$ is not available except at $\xi = 0$.

Making use of the fact that

$$\begin{aligned} \dot{y}_c(t + \xi) &= f\{t + \xi, y_T(t + \xi) + [y_c(t + \xi) - y_T(t + \xi)]\} \\ &= \dot{y}_T(t + \xi) + O(\xi^3) \end{aligned} \tag{2.62}$$

we obtain

$$\begin{aligned} E_0 &= \int_0^h [\dot{y}_c(t + \xi) - \dot{y}_e(t + \xi)] d\xi + \int_0^h [\dot{y}_T(t + \xi) - \dot{y}_c(t + \xi)] d\xi \\ &= \int_0^h [\dot{y}_c(t + \xi) - \dot{y}_e(t + \xi)] d\xi + O(h^4) \end{aligned}$$

If $\ddot{y}_c(t + \xi) - \ddot{y}_e(t + \xi)$ does not change sign for $0 \leq \xi \leq h$,

$$|E_0| \leq h|\dot{y}_T(t + h) - \dot{y}_e(t + h)| \leq h|\dot{y}_c(t + h) - \dot{y}_e(t + h)| + |O(h^4)|.$$

Thus, for small h ,

$$E = h|\dot{y}_c(t + h) - \dot{y}_e(t + h)|$$

yields an estimate of the truncation error.

Let u_1, u_2, l_1, l_2 be real numbers ≥ 0 . Compute

$$U = u_1 + u_2 |y_c(t + h)|$$

$$L = l_1 + l_2 |y_c(t + h)|.$$

- If $|E| > U$, the step size is reduced.
- If $|E| < L$, the step size is increased.
- If $L \leq E \leq U$, the step size is not altered.

For absolute error control, set

$$u_1 = 0.0075 \quad u_2 = 0 \quad l_1 = 0.0002 \quad l_2 = 0.$$

For relative error control set

$$u_1 = 0.005 \quad u_2 = 0.005 \quad l_1 = 0.0001 \quad l_2 = 0.0002.$$

General Absolute and Relative Error Criteria

All three explicit integration methods described in this section are approximate methods that are used to integrate differential equations and, thereby, update the state variables at each time step. Each of these methods has an automatic control that allows step size to be increased and decreased during the transient problem. The method of control used compares some function of the step size and the derivatives of the state variables to a quantity that serves as a standard as, for example, in Equation (2.49). The standard may be a constant or a variable function of the state variable. If the former is used, it is termed an *absolute error criterion*; if the latter is used, it is termed a *relative error criterion*.

Consider the situation in which two state variables are very different in size. Let $y_1(t) = 1$ and $y_2(t) = 100$. For all practical purposes, it is usually unnecessary to integrate the derivatives of these state variables to the same accuracy. If an absolute error criterion is used, this is just what is done, and the step size may be unnecessarily inhibited. If, however, a relative error criterion is used, effectively a larger error will be tolerated for the integration of $y_2(t)$ and a larger step size will be permitted. In general, then, it would be best for the user to use relative error control when large values of state variables (capacitor voltages and inductor currents) are expected.

Relative error control is programed with all three explicit methods, but the user may easily modify the relative controls or enter absolute controls (see [2, section RUN CONTROLS]). If larger numbers are used for the u_1 or u_2 entries, the solution process will be less likely to halve any particular solution step size; smaller numbers would increase the likelihood of reduced step sizes. If larger numbers are used for the l_1 and l_2 entries, the solution process is more likely to increase any particular solution step size; smaller numbers make increased step sizes less likely. The user should realize that any increase in solution speed that may result from adjustment of these numbers must necessarily come at the expense of integration accuracy.

Implicit Method

The integration methods described in subsections 2.3.5 [RUK Formulas and Variable Step Size Control, Modified Trapezoidal Integration (TRAP) and Exponential Integration (XPO)] are all explicit in form and can be used interchangeably within the mathematical formulation of SCEPTRE without difficulty. If, however, any implicit method is to be used, whether single step or multistep, an additional computational step is necessary. This step is discussed in the following paragraphs and in Appendix A.

Basic Implicit Format A generalized form of all implicit integration methods can be written as

$$Y(n+1) = \sum_{i=0}^p a_i Y(n-i) + h \sum_{i=-1}^p b_i \dot{Y}(n-i). \quad (2.63)$$

where Y is the vector of system state variables, $\dot{Y}$ is the vector of state variable derivatives, n is the step number, h is the step size and the a_i, b_i are suitably chosen constants. Multistep methods are introduced if $p > 0$. The simplest derivative form of Equation (2.63) is

$$Y(n+1) = Y(n) + h\dot{Y}(n+1).$$

which is commonly referred to as the implicit or backward Euler technique. If an m th order system of differential equations is to be solved by this method, the following generalized matrix equation will result:

$$\begin{bmatrix} 1 - hg \frac{\partial \dot{Y}_1}{\partial Y_1}(Y_1, \dots, Y_m, t) & \dots & -hg \frac{\partial \dot{Y}_1}{\partial Y_m}(Y_1, \dots, Y_m, t) \\ \vdots & & \vdots \\ -hg \frac{\partial \dot{Y}_m}{\partial Y_1}(Y_1, \dots, Y_m, t) & \dots & 1 - hg \frac{\partial \dot{Y}_m}{\partial Y_m}(Y_1, \dots, Y_m, t) \end{bmatrix} \begin{bmatrix} \Delta Y_1^k \\ \vdots \\ \Delta Y_m^k \end{bmatrix} = \begin{bmatrix} -F_1(Y_1, \dots, Y_m, t) \\ \vdots \\ -F_m(Y_1, \dots, Y_m, t) \end{bmatrix}. \quad (2.64)$$

The iteration implied in Equation (2.64) is carried out to convergence at each time step. The k superscripts here indicate the k th approximation to the final value at convergence at each step, g is a constant that depends on the order of integration, and F_i is a function of the i th differential equation, the step size and past values of the m th state variable. Sparse matrix techniques will be applied to the operation implied by Equation (2.64) when large problems are encountered.

The Jacobian The Y_i terms in Equation (2.64) are readily available from the basic program formulation; but the general partial derivative term

$$\frac{\partial \dot{Y}_i}{\partial Y_j}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq m$$

is not. To determine what is really needed, it is desirable to frame the entire derivation of the symbolic Jacobian in terms of the general matrix equation

$$\dot{Y} = AY + BU + N\dot{U}. \quad (2.65)$$

Since the desired quantities are the general $\partial \dot{Y}_i / \partial Y_j$ it can be seen from partial differentiation of Equation (2.65) that these are contained in the matrix A. Hence the convenience of the general notation is given in Equation (2.65).

What now follows is the construction in symbolic form of the general matrix A in terms of the mathematical formulation that was derived in subsection 2.3.1. Begin with

$$C_{44}\dot{V}_4 = I_4 = B_{14}^T I_1 + B_{24}^T I_2 + B_{34}^T I_3 + B_{84}^T J_8 \quad (2.5)$$

and

$$V_3 = -B_{34}V_4 - B_{35}V_5 - B_{36}V_6 - B_{37}E_7 \quad (2.3).$$

Substitute Equation (2.9) and (2.13) into Equation (2.5) for

$$\begin{aligned} C_{44}\dot{V}_4 &= B_{14}^T C_{11} (-B_{14}\dot{V}_4 - B_{17}\dot{E}_7) \\ &+ B_{24}^T M_R^{-1} (-B_{24}V_4 - B_{27}E_7 - B_{25}R_{55}B_{35}^T I_3 - B_{25}R_{55}B_{85}^T J_8) \\ &+ B_{34}^T I_3 + B_{84}^T J_8. \end{aligned} \quad (2.66)$$

Use Equations (2.24), (2.10), (2.23) and (2.15) into Equation (2.3) for

$$\begin{aligned} L_{33}\dot{I}_3 + L_{36} (B_{36}^T \dot{I}_3 + B_{86}^T \dot{J}_8) &= \\ &- B_{34}V_4 - B_{35}M_G^{-1} \{-B_{25}^T G_{22}(B_{24}V_4 + B_{27}E_7) + B_{35}^T I_3 + B_{85}^T J_8\} \\ &- B_{36}(L_{63} + L_{66}B_{36}^T)\dot{I}_3 - B_{36}L_{66}B_{86}^T \dot{J}_8 - B_{37}E_7. \end{aligned} \quad (2.67)$$

Equation (2.66) can be manipulated to yield

$$\begin{aligned} (C_{44} + B_{14}^T C_{11} B_{14}) \dot{V}_4 &= -B_{14}^T C_{11} B_{17} \dot{E}_7 \\ &\quad - B_{24}^T M_R^{-1} (B_{24} V_4 + B_{27} E_7 + B_{25} R_{55} B_{35}^T I_3 + B_{25} R_{55} B_{85}^T J_8) \\ &\quad + B_{34}^T I_3 + B_{84}^T J_8 . \end{aligned} \quad (2.68)$$

Equation (2.67) yields

$$\begin{aligned} [L_{33} + L_{36} B_{36}^T + B_{36} L_{63} + B_{36} L_{66} B_{36}^T] \dot{I}_3 &= -B_{34} V_4 \\ &\quad - B_{35} M_G^{-1} (-B_{25}^T G_{22} B_{24} V_4 - B_{25}^T G_{22} B_{27} E_7 + B_{35}^T I_3 + B_{85}^T J_8) \\ &\quad - B_{37} E_7 - L_{36} B_{86}^T \dot{J}_8 - B_{36} L_{66} B_{86}^T \dot{J}_8 . \end{aligned} \quad (2.69)$$

If only the coefficients of $V_4, I_3, \dot{V}_4, \dot{I}_3$ (the state variables and the state variable derivatives) are retained, Equations (2.66) and (2.67) become considerably simplified and can be written in matrix notation as

$$\begin{bmatrix} \dot{V}_4 \\ \dot{I}_3 \end{bmatrix} = \begin{bmatrix} M_C & 0 \\ 0 & M_L \end{bmatrix}^{-1} \begin{bmatrix} (-B_{24}^T M_R^{-1} B_{24}) & (B_{34}^T - B_{24}^T M_R^{-1} B_{25} R_{55} B_{35}^T) \\ (B_{35} M_G^{-1} B_{25}^T G_{22} B_{24} - B_{34}) & (-B_{35} M_G^{-1} B_{35}^T) \end{bmatrix} \begin{bmatrix} V_4 \\ I_3 \end{bmatrix} \quad (2.70)$$

where

$$M_C = C_{44} + B_{14}^T C_{11} B_{14}$$

and

$$M_L = L_{33} + L_{36} B_{36}^T + B_{36} L_{63} + B_{36} L_{66} B_{36}^T .$$

The matrix appearing on the right side of Equation (2.70) is now in a form corresponding to the first term on the right side of Equation (2.65). This matrix is the symbolic form of the general A matrix that is needed to use implicit integration ([17, 18]) with the basic program formulation.

Another method is available to construct the Jacobian that is based on a numerical rather than a symbolic approach. Each time that the Jacobian is to be re-evaluated, m calls are made to SIMUL8 to compute

$$\dot{Y}^{(j)} = F(Y_1, Y_2, \dots, Y_{j-1}, Y_j + \Delta Y_j, Y_{j+1}, \dots, Y_m, t) .$$

Approximations to the desired partials are then obtained by

$$\frac{\partial \dot{Y}_i}{\partial Y_j} \approx \frac{\dot{Y}_i^{(j)} - \dot{Y}_i}{\Delta Y_j} .$$

2.4 DC Solutions

SCEPTRE offers five DC solution modes, Initial Conditions, Sensitivity, Monte Carlo, Worst-Case and Optimization. Any one of the five may be used alone, or as a source of initial conditions for a Transient or AC run. Alternatively, the user may supply initial conditions data himself as entries to a transient or AC run without the help of the DC options. One of the DC options, called Initial Conditions, takes part in all DC solutions. It may be called separately, and if any other DC option is called, that option uses two or more passes of the Initial Conditions calculation to produce its results. The Initial Conditions solution (and thus the other DC options also) may use either the Newton-Raphson or implicit method (see [2, Appendix A]).

2.4.1 Initial Conditions Solution by Newton-Raphson Method

Technique Description

Many practical circuits require computer solution of the initial conditions prevailing at the start of the transient (time = t_0) before the transient solution can begin. These values can always be determined by a separate transient run in which all forcing functions are held at the values for time = t_0 . However, an alternate procedure based on an iteration technique using independent variables other than time was considered desirable. This procedure presents the advantage of economy of machine time on all circuits for which convergence occurs. This section will describe the formulation of this portion of SCEPTRE, which is completely independent of the transient formulation.

A similar derivation for the adjoint calculation is given in Appendix F.

If an L tree is set up based on the preference order E, L, R, C a general B matrix may be set up according to the procedure outlined in subsection 2.2. The zero-valued submatrices arise from the L tree and the preference order⁵ The resulting B matrix is shown in Table 2.4. The following equations

Table 2.4: General B Matrix from L Tree

	4	5	6	7
1	B_{14}	B_{15}	B_{16}	B_{17}
2	0	B_{25}	B_{26}	B_{27}
3	0	0	B_{36}	B_{37}
8	B_{84}	B_{85}	B_{86}	B_{87}
9	B_{94}	B_{95}	B_{96}	B_{97}
0	B_{04}	B_{05}	B_{06}	B_{07}

(among others) arise from this B matrix if vectors V_6, I_1, I_4 and submatrix B_{94} are assumed to be zero. These assumptions are based on the known final values of V_6, I_1, I_4 for the initial condition problem and the absence of any current-source capacitor cut sets (see the restrictions in subsection 2.4.1).

$$V_9 + B_{95}V_5 + B_{97}E_7 = 0 \quad (2.71)$$

$$V_2 + B_{25}V_5 + B_{27}E_7 = 0 \quad (2.72)$$

$$I_5 - B_{25}^T I_2 - B_{85}^T J_8 - B_{95}^T J_9 - B_{05}^T J_0 = 0 \quad (2.73)$$

To get all the variables of Equation (2.71) in terms of V_9, V_2, V_5 , and effective sources, the following substitutions are made:

$$I_5 = G_{55}V_5$$

$$I_2 = G_{22}V_5$$

$$J_0 = \alpha J_9$$

$$J_9 = G_{99}V_9 + Q_9.$$

where

⁵For example, the submatrix B_{24} must always be zero since non-zero entries in it could only arise when resistor links close loops containing capacitor tree branches. The preference order prohibits this possibility.

- α is a matrix containing the number of rows equal to the number of secondary current sources and the number of columns equal to the number of primary current sources
- G_{55} and G_{22} are diagonal matrices containing only conductances
- G_{99} is a diagonal matrix containing only diode and transistor junction conductances
- Q terms are described in Appendix B.

Then, Equation (2.73) becomes

$$G_{55}V_5 - B_{25}^T G_{22}V_2 - B_{85}^T J_8 - [B_{95}^T + B_{05}^T \alpha] [G_{99}V_9 + Q_9] = 0. \quad (2.74)$$

Equations (2.71), (2.72), and (2.74) may be designated as $F_1(V_9, V_2, V_5)$, $F_2(V_9, V_2, V_5)$ and $F_3(V_9, V_2, V_5)$, respectively. If the basic Newton-Raphson method (see Appendix D) is applied to Equations (2.71), (2.72), and (2.74):

$$\begin{aligned} F_1(V_9, V_2, V_5) + \frac{\partial F_1}{\partial V_9}(V_9, V_2, V_5)\Delta V_9 + \frac{\partial F_1}{\partial V_2}(V_9, V_2, V_5)\Delta V_2 + \frac{\partial F_1}{\partial V_5}(V_9, V_2, V_5)\Delta V_5 &= 0 \\ F_2(V_9, V_2, V_5) + \frac{\partial F_2}{\partial V_9}(V_9, V_2, V_5)\Delta V_9 + \frac{\partial F_2}{\partial V_2}(V_9, V_2, V_5)\Delta V_2 + \frac{\partial F_2}{\partial V_5}(V_9, V_2, V_5)\Delta V_5 &= 0 \\ F_3(V_9, V_2, V_5) + \frac{\partial F_3}{\partial V_9}(V_9, V_2, V_5)\Delta V_9 + \frac{\partial F_3}{\partial V_2}(V_9, V_2, V_5)\Delta V_2 + \frac{\partial F_3}{\partial V_5}(V_9, V_2, V_5)\Delta V_5 &= 0 \end{aligned}$$

or

$$\begin{bmatrix} F_1(V_9, V_2, V_5) \\ F_2(V_9, V_2, V_5) \\ F_3(V_9, V_2, V_5) \end{bmatrix} + Z \begin{bmatrix} \Delta V_9 \\ \Delta V_2 \\ \Delta V_5 \end{bmatrix} = 0 \quad (2.75)$$

where the Jacobian is

$$Z = \begin{bmatrix} \frac{\partial F_1}{\partial V_9}(V_9, V_2, V_5) & \frac{\partial F_1}{\partial V_2}(V_9, V_2, V_5) & \frac{\partial F_1}{\partial V_5}(V_9, V_2, V_5) \\ \frac{\partial F_2}{\partial V_9}(V_9, V_2, V_5) & \frac{\partial F_2}{\partial V_2}(V_9, V_2, V_5) & \frac{\partial F_2}{\partial V_5}(V_9, V_2, V_5) \\ \frac{\partial F_3}{\partial V_9}(V_9, V_2, V_5) & \frac{\partial F_3}{\partial V_2}(V_9, V_2, V_5) & \frac{\partial F_3}{\partial V_5}(V_9, V_2, V_5) \end{bmatrix} \quad (2.76)$$

If

$$\begin{aligned} \Delta V_9 &= V_9(n+1) - V_9(n) \\ \Delta V_2 &= V_2(n+1) - V_2(n) \\ \Delta V_5 &= V_5(n+1) - V_5(n) \end{aligned}$$

where n is used to designate the results of the n th iteration pass, then (2.75) becomes

$$\begin{bmatrix} F_1(V_9, V_2, V_5) \\ F_2(V_9, V_2, V_5) \\ F_3(V_9, V_2, V_5) \end{bmatrix} + Z \begin{bmatrix} V_9(n+1) - V_9(n) \\ V_2(n+1) - V_2(n) \\ V_5(n+1) - V_5(n) \end{bmatrix} = 0 \quad (2.77)$$

leading directly to

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} V_9(n) \\ V_2(n) \\ V_5(n) \end{bmatrix} - Z^{-1} \begin{bmatrix} F_1(V_9, V_2, V_5) \\ F_2(V_9, V_2, V_5) \\ F_3(V_9, V_2, V_5) \end{bmatrix}. \quad (2.78)$$

Equation (2.76) may be written more explicitly if the indicated differentiations are performed on Equations (2.71), (2.72), and (2.74) to obtain:

$$Z = \begin{bmatrix} I & 0 & B_{95} \\ 0 & I & B_{25} \\ -[B_{95}^T + B_{05}^T \alpha] G_{99} & -B_{25}^T G_{22} & G_{55} \end{bmatrix}$$

so that Equation (2.78) becomes

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} V_9(n) \\ V_2(n) \\ V_5(n) \end{bmatrix} - \begin{bmatrix} \\ \\ Z \end{bmatrix}^{-1} \begin{bmatrix} F_1(V_9, V_2, V_5) \\ F_2(V_9, V_2, V_5) \\ F_3(V_9, V_2, V_5) \end{bmatrix}. \quad (2.79)$$

Equation (2.79) may be written in more convenient form as follows (see also Appendix E):

$$\begin{bmatrix} V_9(n+1) \\ V_2(n+1) \\ V_5(n+1) \end{bmatrix} = \begin{bmatrix} I & 0 & B_{95} \\ 0 & I & B_{25} \\ -[B_{95}^T + B_{05}^T \alpha] G_{99} & -B_{25}^T G_{22} & G_{55} \end{bmatrix}^{-1} \begin{bmatrix} -B_{97} E_7 \\ -B_{27} E_7 \\ [B_{95}^T + B_{05}^T \alpha] Q_9 + B_{85}^T J_8 \end{bmatrix}. \quad (2.80)$$

Equation (2.80) signifies a computational sequence as follows: Quantities V_9 , V_2 , and V_5 , the vectors of voltages across primary current sources, resistor links, and resistor tree branches respectively, each have some assumed value (usually zero) to begin the computation at $n = 0$. All members of the right side of Equation (2.80) that may be voltage dependent are updated. The left side of Equation (2.80) is then computed and the first iteration ($n = 1$) is complete. The right side of Equation (2.80) is re-evaluated on the basis of the results of the first iteration and the left side is again computed, thereby completing the second iteration ($n = 2$). This process is repeated up to 100 times. After any iteration, if

$$|V_{n+1} - V_n| \leq |V_{n+1}| + 0.0001 \quad (2.81)$$

is satisfied for the set of all voltages in Equation (2.79), convergence is considered to have occurred and the procedure is terminated. If after 100 iterations Equation (2.81) is not satisfied, a diagnostic message will be printed indicating that convergence has not occurred. Experience to date has shown that the Newton-Raphson procedure will converge for most circuits in less than 30 iterations.

Convergence of the Newton-Raphson method can sometimes be prohibitively delayed if for some reason a large forward bias ($V > 0.8$ V) is applied to any diode or transistor junction that has been represented by the user in the conventional closed form $J = I_s(e^{\Theta V} - 1)$. In this case, the slope of the diode curve, given by $G_J = \frac{dJ}{dV} = \Theta I_s e^{\Theta V}$, will contribute a very large term in G_{99} and consequently in the Z matrix. The practical results of this can be more easily appreciated by consideration of a one equation system as represented by the second equation in Appendix D. A large derivative caused by a highly forward-biased diode leads to a very small step (Δx) in the independent variable. Many steps will therefore be required to complete convergence.

If convergence has occurred, quantities V_9, V_2, V_5 are now known. There remains to compute only capacitor voltages and inductor currents since capacitor currents and inductor voltages must be zero.

Computing Capacitor Voltages

Form the B matrix in Table 2.4, the capacitor link voltages can be written in terms of the tree branch voltages as

$$V_1 = -B_{14}V_4 - B_{15}V_5 - B_{17}E_7. \quad (2.82)$$

In addition the principle of conservation of charge permits

$$-B_{14}^T C_{11} V_1 + C_{44} V_4 = -B_{14}^T C_{11} V_1(0) + C_{44} V_4(0) \quad (2.83)$$

where $V_1(0)$ and $V_4(0)$ are initial voltages that may be specified by the user. Equation (2.82) and (2.83) lead to

$$V_4 = M_c^{-1} \{-B_{14}^T C_{11} B_{15} V_5 - B_{14}^T C_{11} B_{17} E_7 - B_{14}^T C_{11} V_1(0) + C_{44} V_4(0)\} \quad (2.84)$$

where $M_c = B_{14}^T C_{11} B_{14} + C_{44}$. Once V_4 is determined, V_1 may be found from Equation (2.82).

Note that Class-4 elements can occur only if a capacitor cut set exists. Otherwise Equation (2.84) will be identically zero and all capacitor voltages can be determined from Equation (2.82).

The use of Equation (2.83) permits solution of a class of networks in which the final value of capacitor voltage is dependent on an initial value. Consider the circuit shown in figure 2.7, which contains a capacitor cut set. If both capacitors are initially uncharged, the final values of the capacitor voltages

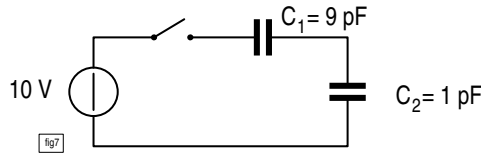


Figure 2.7: Capacitor Cut Set Circuit

after the switch is closed must be $V_{C1} = 1$ volts, $V_{C2} = 9$ volts. If however, capacitor C_1 has an initial charge of 5 volts, the principle of conservation of charge (reflected in equation (2.83)) requires the final result to be $V_{C1} = 5.5$ volts, $V_{C2} = 4.5$ volts.

Computing Inductor Currents

Since V_2 is known

$$I_2 = R_{22}^{-1} V_2 \quad (2.85)$$

the inductor tree branch currents than can be written from the B matrix in terms of link currents as

$$I_6 = B_{26}^T I_2 + B_{36}^T I_3 + B_{86}^T J_8 + B_{96}^T J_9 + B_{06}^T J_0. \quad (2.86)$$

In addition, the flux relations around on inductor loop permit

$$B_{36} L_{66} I_6 + L_{33} I_3 = B_{36} L_{66} I_6(0) + L_{33} I_3(0) \quad (2.87)$$

where $I_6(0)$ and $I_3(0)$ are initial currents that may be specified by the user. Equations (2.86) and (2.87) lead to

$$I_3 = M_L^{-1} \{L_{33} I_3(0) + B_{36} L_{66} I_6(0) - B_{36} L_{66} [B_{26}^T I_2 + B_{86}^T J_8 + B_{96}^T J_9 + B_{06}^T J_0]\} \quad (2.88)$$

where $M_L = B_{36}L_{66}B_{36}^T + L_{33}$.

Once I_3 is determined, I_6 may be found from Equation (2.86). Note that Class-3 elements can occur only if an inductor loop exists.⁶ Otherwise, Equation (2.88) will be identically zero and all inductor currents can be determined from equation (2.86).

The use of Equation (2.88) permits the solution of a class of networks in which the final values of inductor currents are dependent on the sizes of the respective inductances. Consider the circuit shown in figure 2.8, which contains an inductor loop. If both inductors are initially relaxed, the final

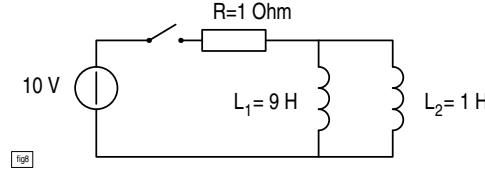


Figure 2.8: Inductor Loop Circuit

values of the inductor currents after the switch is closed must be $I_{L1} = 1$ A, $I_{L2} = 9$ A.

The task of computing the capacitor voltages and inductor currents complete the initial-conditions problem. These quantities may then be transferred to the transient program to serve as initial conditions.

Restrictions

Since the initial-conditions program is formulated differently and for a different purpose than is the transient program, certain restrictions apply to the initial-conditions program that do not apply to the transient program. These restrictions and the practical considerations that lead to them are as follows:

1. No circuit containing a loop composed entirely of voltage sources and inductors will be accommodated. This situation would cause an infinite inductor current and is obviously of no practical importance. The presence of an E-L loop is disclosed by the condition $B_{37} \neq 0$.
2. No circuit containing a cut set composed entirely of current sources and capacitors will be accommodated (see figure 2.9). This situation would invalidate equation (2.71) and complicate

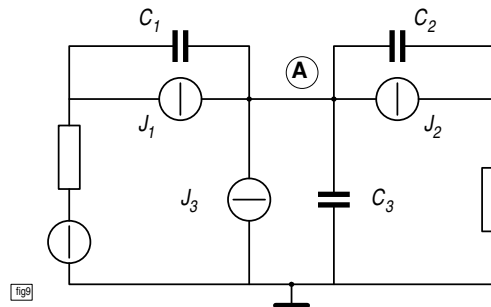


Figure 2.9: A Current-Source Capacitor Cut Set

the solution process by requiring that V_4 be carried along in the Newton-Raphson procedure.

⁶See paragraph (subsequent subsection 2.4.1) on network restrictions for qualification of this.

This cut set situation can always be removed by arbitrarily connecting a large resistor from node A to the ground. Note that the configuration of figure 2.9 could be handled by the transient portion of SCEPTRE if a Newton-Raphson solution was not desired. The presence of a J-C cut set is disclosed by the condition that either B_{84}, B_{94} or $B_{04} \neq 0$.

3. The choice of independent variables will be somewhat restricted. No resistor or inductor current may be used as an independent variable. Furthermore, a capacitor voltage may be used as an independent variable only if it is in parallel with a resistor or current source ($V_C = V_R, V_C = V_J$). The objective here is to avoid the need for any auxiliary computation between passes of the iteration procedure.
4. Networks containing only capacitor cut sets can be accommodated only if the members of the cut set are constant.
5. Networks containing only inductor loops can be accommodated only if the members of the loop are constant.

2.4.2 Monte Carlo

Monte Carlo calculations in SCEPTRE consist of multiple initial conditions solutions for which certain element values have been chosen using probability distributions specified by the user. These variable elements, representing perhaps production deviations from nominal values, cause corresponding variations in circuit output quantities. The Monte Carlo function defines the selection of variable element values, controls the corresponding iterative solutions, monitors output values, and tabulates statistics for the output quantities.

The user can specify either the uniform or Gaussian distribution for the variable elements. Since the solution is calculated for $\text{TIME} = 0$, it is independent of capacitive and inductive elements. Thus, it is only useful to vary resistors, independent current and voltage sources, and primary and secondary current sources specified by defined parameters.

The value of a variable element is determined by its specified mean, μ , standard deviation, σ , and a random number, ζ , picked from the selected probability distribution. The random number generator provides samples from the selected type of distribution with zero mean and unit standard deviation. Then the element value e.g. R , is calculated as

$$R = \mu + \zeta \cdot \sigma .$$

If the random behavior of the element value cannot be directly defined in terms of one of the two standard distributions, the mechanism of defined parameters may be used. By specifying the means and standard deviations for one or more defined parameters, and defining the element value using an equation, expression or table in terms of these parameters, the user can obtain a more complicated probabilistic behavior of the element value.

As each new set of values is chosen for the random element value, an initial conditions calculation is executed. For each output quantity, the sum and sum of squares are accumulated, and the minimum and maximum observed over all iterations are tabulated. The mean and standard deviation for each output X are then computed as

$$M_x = \frac{\sum_i x_i}{N}$$

and

$$S_x = \sqrt{\left[\frac{\sum_i (x_i - M_x)^2}{N - 1} \right]}$$

where M_x is the mean, S_x is the standard deviation, and the sums are taken from 1 to N, where N is the number of Monte Carlo iterations.

2.4.3 Sensitivity

The Sensitivity option gives the user the partial derivative of a network function with respect to a list of network independent variables. This option gives the user the "un-normalized sensitivity", i.e.

$$\int_X^{UH} \frac{\partial H}{\partial X}$$

The normalized sensitivity, or the percentage change in a function with respect to the percentage change in an independent variable, may be obtained, if desired, by using a defined parameter. The normalized sensitivity is

$$\int_X^H \frac{X}{H} \frac{\partial H}{\partial X}.$$

To calculate sensitivity, an initial conditions run is made on the original network, and appropriate network element currents and voltages are saved. An adjoint network as described in Appendix F is generated [19, 20]. For each network function whose sensitivity is requested, an initial conditions run is made on the adjoint network with a proper source, and the element currents and voltages in the adjoint run are saved. The proper product of the original network values and the adjoint network values (see Appendix F) gives the required sensitivity. Table 2.5 gives the sensitivity (superscript "a" indicates adjoint values).

Table 2.5: Sensitivity Computations

Independent Variable	Sensitivity
Resistor R	(Current through the resistor in original IC run) · (Current through the same resistor in adjoint IC run), i.e., $(I_R) \cdot (I_R^a)$
Independent Voltage Source E_7	Negative current through the voltage source in adjoint IC run, i.e., $(-I_{E7}^a)$
Independent Current Source J_8	Negative Voltage across the current source in adjoint IC run, i.e., $(-V_{J8}^a)$
Factor in Secondary Current Source given by Defined Parameter P where $J_0 = P * J_9$	(The primary current in original IC run) · (Negative voltage across the secondary source in adjoint IC run), i.e., $(J_9) \cdot (-V_{J0}^a)$

2.4.4 Worst Case

A Worst-Case calculation extrapolates linearly along the gradient vector of an objective function, F , to reach a function minimum and maximum at one or more independent variable bounds. The gradient vector $\vec{\nabla}F$ is computed⁷ for nominal circuit conditions, and it is assumed that F depends linearly on the N independent variables. The gradient indicates the direction in which the function grows most rapidly, and the assumed linearity allows extrapolation along this vector to reach the minimum and maximum.

The maximum of F is then obtained by extrapolating the independent variable values $\vec{X}$ as far as possible along the gradient vector:

$$\vec{X}_H = \vec{Y} + \lambda \vec{\nabla}F, \quad \lambda > 0. \quad (2.89)$$

Here, $\vec{X}_H$ is the independent variable vector producing the high value of F , $\vec{Y}$ is the vector of independent variables at the nominal point, and λ is taken as large as possible with $\vec{X}_H$ remaining within bounds.

Under the same assumption, the minimum of F is obtained by extrapolation along the negative of the gradient vector:

$$\vec{X}_L = \vec{Y} - \mu \vec{\nabla}F, \quad \mu > 0. \quad (2.90)$$

$\vec{X}_L$ is the vector of independent variables at the point producing the low value of F , and μ is chosen as large as possible, keeping the independent variables within bounds.

The independent variables are specified with lower and upper bounds

$$P_i \leq X_i \leq Q_i, \quad i = 1 \dots N.$$

If the i th component of the gradient vector $\partial F / \partial X_i$ is positive, then F increases as X_i increases from Y_i toward Q_i . If, however, $\partial F / \partial X_i$ is negative, F increases as X_i decreases from Y_i toward P_i .

The general equation of a line in N dimensional space may be written as

$$\frac{X_1 - Y_1}{\mu_1} = \frac{X_2 - Y_2}{\mu_2} = \dots = \frac{X_N - Y_N}{\mu_N}$$

where $(X_i, i = 1 \dots N)$ and $(Y_i, i = 1 \dots N)$ are two points on the line and μ is a unit vector in the direction of the line.

The equation can be restated

$$\frac{X_i - Y_i}{\mu_i} = \text{constant}. \quad (2.91)$$

When the j th component takes on its lower bounding value, we have

$$\frac{X_i - Y_i}{\mu_i} = \frac{P_j - Y_j}{\mu_j}, \quad i = 1 \dots N$$

or,

$$X_i = \frac{\mu_i}{\mu_j} (P_j - Y_j) + Y_i.$$

⁷See Appendix F for a discussion of the Adjoint method, used for computing the gradient vector.

The length of the line segment is given by

$$D_{P_j}^2 = \sum_i (X_i - Y_i)^2 = \frac{(P_j - Y_j)^2}{\mu_j^2} \sum_i \mu_i^2 = \frac{(P_j - Y_j)^2}{\mu_j^2},$$

since μ is a unit vector. Thus,

$$D_{P_j} = \left| \frac{P_j - Y_j}{\mu_j} \right|,$$

the distance along the gradient to the lower bounding surface of the j th independent variable.

Similarly,

$$D_{Q_j} = \left| \frac{Q_j - Y_j}{\mu_j} \right|,$$

the distance to the corresponding upper boundary surface.

For the high limit calculation, a search is made through all D_{P_i} for which $\partial F / \partial X_i$ is positive, and all D_{Q_i} for which $\partial F / \partial X_i$ is negative, until the smallest number from this set is found. (Note that F is independent of any X_i for which $\partial F / \partial X_i = 0$). This minimum value, D_{HIGH} represents the largest distance $\vec{X}$ can move in the direction of the gradient vector before it reaches a boundary. X_{HIGH} is the independent variable which established the allowable displacement of $\vec{X}$ associated with D_{HIGH} .

A similar search through all D_{P_i} for which $\partial F / \partial X_i$ is negative, and all D_{Q_i} for which $\partial F / \partial X_i$ is positive, produces another minimum, D_{LOW} , which is the largest possible displacement of $\vec{X}$ from $\vec{Y}$ along the negative of the gradient vector before striking boundary. X_{LOW} is the independent variable which established the allowable displacement associated with D_{LOW} .

X_{HIGH} and X_{LOW} may or may not be the same variable. Figure 2.10 illustrates a two-dimensional case where the high limit is determined by the upper bound of one variable (X_2), and the low limit is determined by the upper bound of another (X_1).

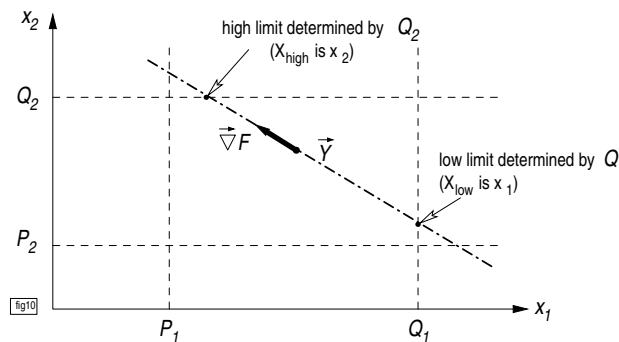


Figure 2.10: Two-Dimensional Example of Worst-Case Limits

When the distance D_{HIGH} has been determined, Equation (2.91) can be used to define all independent variable values, $\vec{X}_H$, at the high limit:

$$\frac{X_i - Y_i}{\mu_i} = D_{\text{HIGH}}$$

or

$$\vec{X}_H = Y + \mu D_{\text{HIGH}} .$$

Note that this is exactly in the form of Equation (2.89), since $\vec{\mu}$ is proportional to $\vec{\nabla}F$ and $D_{\text{HIGH}} > 0$. In a similar way, we find the independent variables at the low limit to be

$$X_L = Y - \mu D_{\text{LOW}}$$

as required by Equation (2.90).

When X_H and X_L have been found, computations of objective function value and gradient at each of the two points are carried out. Inspection of the gradient vector at these points can give information on whether the true extreme values occur within the bounded area or if they are external.

2.4.5 Optimization

The Optimization option finds the local minimum of an objective function with respect to a list of bounded independent variables. The local minimum is the point in the independent variable space at which each component of the gradient either is zero, or is negative (function decreasing) with its corresponding independent variable at a bound.

The Optimization, or more accurately minimization, function is based on the *Davidon algorithm* [21]. As in most such processes, the gradient is computed for some set of values of the independent variables, $\vec{Y}$, within all boundaries, and the independent variable vector is displaced along the gradient. At this new point, the gradient vector is again computed, and the cycle repeats until the error (discussed below) is less than some criterion value.

The adjoint method of computing the gradient obviates the use of a direct search method with its inherent numerical problems. Furthermore, since the adjoint procedure determines the gradient with only two IC solutions regardless of the number of independent variables, it has a speed advantage over direct search methods and other gradient methods. This speed advantage is enhanced because the auxiliary IC solution (for the adjoint network) is strictly linear and will generally converge quickly. Thus the combination of rapid gradient calculation and efficient search for the function minimum provide the SCEPTRE user with an extremely versatile tool.

The Davidon method is a synthesis of two techniques: The *Steepest Descent Method* and the *Newton-Raphson Method* applied in the Initial Conditions calculation (subsection 2.4.1). The Newton-Raphson method is based on the truncated Taylor expansion

$$F(\vec{X}) = F(\vec{Y}) + \vec{h} \cdot \vec{\nabla}F(\vec{Y}) + \frac{1}{2} \vec{h} \cdot G(\vec{Y}) \vec{h}$$

where $\vec{h}$ is the displacement vector ($\vec{X} - \vec{Y}$) and G is the Hessian matrix of second partial derivatives of the objective function:

$$G_{ij} = \frac{\partial^2 F}{\partial X_i \partial X_j} .$$

Differentiation of this approximation to $F(\vec{X})$ leads to the expression for the gradient:

$$\vec{\nabla}F(\vec{X}) = \vec{\nabla}F(\vec{Y}) + G(\vec{Y}) \vec{h} .$$

At the minimum value $\vec{X}$, $\vec{\nabla}F(\vec{X}) = 0$, so that the displacement $\vec{h}$ is given by

$$\vec{h} = -\frac{1}{G(\vec{Y})}\vec{\nabla}F(\vec{Y}) .$$

In the vicinity of the minimum, it is assumed that F is essentially a quadratic function of $\vec{X}$, so that the point at which G is evaluated is not critical.

The Steepest Descent step is determined completely by the gradient vector, and thus has the advantage of simplicity. However, the process is very inefficient, even for functions with quadratic dependence on the independent variables, and is not generally recommended except for initiating a minimization calculation.

Because the Davidon technique smoothly changes from the Steepest Descent to the Newton-Raphson iteration, maintaining suitable conditions on h at all times, it is a very effective procedure [22].

The Davidon process [23] starts with an initial approximation H_0 to G^{-1} which is generally the unit matrix and is thus positive definite (since the eigenvalues all have the value unity). The iterations produce an increasingly better approximation H to G^{-1} without matrix inversion, while continuing to enforce the condition of positive definiteness which will be required at the minimum.

The iteration for the Davidon method is given by

$$\vec{X}^{(i+1)} = \vec{X}^{(i)} + h^{(i)} H^{(i)} \vec{\nabla}F^{(i)}$$

where

- $\vec{X}^{(i)}$ is the vector of the independent variables;
- $h^{(i)}$ is the step size, chosen by an auxiliary linear search process;
- $H^{(i)}$ is the approximation to G^{-1}

and the superscript identifies the iteration at which the function is evaluated.

The matrix H is updated as follows. Define the vectors

$$\begin{aligned} \vec{Z} &= -h^{(i)} H^{(i)} \vec{\nabla}F^{(i)} \quad \text{and} \\ \vec{\mu} &= \vec{\nabla}F^{(i+1)} - \vec{\nabla}F^{(i)} \quad \text{then} \\ H^{(i+1)} &= H^{(i)} + A^{(i)} + B^{(i)} \end{aligned}$$

where the elements of matrices $A^{(i)}$ and $B^{(i)}$ are respectively given by

$$a_{jk}^{(i)} = \frac{Z_j Z_k}{\vec{Z} \cdot \vec{\mu}} \tag{2.92}$$

$$b_{jk}^{(i)} = \frac{(H^{(i)} \vec{\mu})_j (H^{(i)} \vec{\mu})_k}{\vec{\mu} \cdot H \vec{\mu}} . \tag{2.93}$$

The matrix A provides convergence of H to G^{-1} , while B maintains the positive definiteness of H at each step.

The implementation in SCEPTRE is based on the original formulation produced by the Argonne National Laboratory [24]. However, the original code was inappropriate for constrained optimization and so the transformation of independent variables suggested by Box [22] was incorporated to properly

take care of the inequality constraints of the SCEPTRE independent variables. This transformation may be stated in the following way: Suppose the i th independent variable X_i is restricted so that

$$P_i \leq X_i \leq Q_i .$$

Then the transformation of variables

$$X_i = \frac{Q_i + P_i}{2} - \frac{Q_i - P_i}{2} \cos Y_i$$

is introduced. The corresponding derivatives is given by

$$\frac{dX_i}{dY_i} = \frac{Q_i - P_i}{2} \sin Y_i .$$

Note that $X_i = P_i \leadsto Y_i = 0, X_i = Q_i \leadsto Y_i = \pi$ and $\frac{dX_i}{dY_i} = 0$ at both extremes. The gradient components with respect to the new variables are given by

$$\frac{\partial F}{\partial Y_i} = \frac{dX_i}{dY_i} \frac{\partial F}{\partial X_i} .$$

Thus, the effect of the transformation is to introduce zeros of the gradient along the boundary, making these points acceptable to the algorithm. No such zeros are introduced within the boundaries, and the algorithm will preferentially converge to an internal minimum if one exists.

Another feature is the removal of a restriction present in the original formulation, which required the minimum value of the objective function to be positive. A floating reference value has been introduced which is smaller than the estimate of the minimum at any stage of the iteration and which is automatically updated as required. The user may specify a reference value known to be smaller than the actual minimum. If he does so, the process will probably converge more quickly, although a poor choice may increase the number of iterations required.

The iteration stops when the change in function value expected from the next increment to the independent variable vector is less than the specified tolerance.

Thus, in figure 2.11, if the anticipated change ΔF due to changing the independent variable to $\vec{X}^{(i+1)}$, computed at point $\vec{X}^{(i)}$ using the supposed quadratic dependence of F on $\vec{X}$, is less than the tolerance, the value $X^{(1+1)}$ will be reported out as the location of the minimum, but no further isolation will be done.

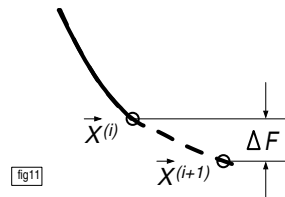


Figure 2.11: Change in ΔF Due to Change in Independent Variable

2.5 AC Solutions

This section describes the AC analysis by SCEPTRE for the following situations:

1. Circuits Containing Independent Source Only (section 2.5.1)

$$\dot{X} = AX + GV, \text{ where } V^T = [E_7 \ J_8]$$

2. Circuits Containing Linearly Dependent Sources (section 2.5.2)

$$\dot{X} = AX + GV + Hu, \text{ where } u^T = [E_Y \ J_X]$$

3. Circuits Requiring Time Derivatives of Independent Sources (section 2.5.3)

$$\dot{X} = AX + GV + Q\dot{V}$$

4. Circuits Containing Linearly Dependent Sources and Requiring Time Derivatives of Independent Sources (section 2.5.4)

$$\dot{X} = AX + GV + Q\dot{V} + Hu$$

Circuits containing linearly dependent sources and requiring time derivatives of both independent and dependent sources are excluded from both transient and AC analysis.

2.5.1 Independent Source Only

State Variable Formulation

Consider the frequency-domain analysis of linear time-invariant systems governed by the equations:

$$\dot{X} = AX + GV \tag{2.94}$$

$$Y = CX + DV \tag{2.95}$$

where

A is an $(N \times N)$ matrix of real coefficients describing the network.

G is an $(N \times N_S)$ matrix of real coefficients relating the externally applied sources to the network.

N is the number of state variables.

N_S is the number of externally applied sources.

N_E is the total number of elements in the network.

N_R is the number of outputs requested, and $N_R \leq 2 \cdot N_E$.

C is an $(N_R \times N)$ matrix of real coefficients relating the state variables to the outputs.

D is an $(N_R \times N_S)$ matrix of real coefficients relating the inputs to the outputs.

X is an $(N \times 1)$ vector composed of state variables.

V is an $(N_S \times 1)$ vector composed of externally applied sources.

Y is an $(N_R \times 1)$ vector composed of requested outputs.

A method which will permit multiple inputs, i.e., multiple penetrations, with frequency dependent magnitude and phase variation. For steady state AC analysis, this requirement on each individual driving function, $1 \leq k \leq N_S$, may be indicated as follows:

$$V_k(t, \omega) = K_k(\omega) e^{\alpha t} \tag{2.96}$$

where

ω is frequency in radians.

α is a complex quantity given by $\alpha = j\omega$.

$K_k(\omega)$ is, in general, a complex function of frequency.

Writing Equation (2.96) in vector notation we have

$$V = K e^{\alpha t} \quad (2.97)$$

where the $(N_S \times 1)$ column vector V is written as N_S terms of the complex frequency variable α with each term having an independent magnitude and phase variation given by K .

The solution of Equation (2.94) is of the form

$$X = X_0 e^{\alpha t} . \quad (2.98)$$

Using Equations (2.97) and (2.98) we can rewrite equation (2.94)

$$\begin{aligned} \frac{d}{dt} (X_0 e^{\alpha t}) &= A (X_0 e^{\alpha t}) + G (K e^{\alpha t}) \\ \alpha X_0 e^{\alpha t} &= A X_0 e^{\alpha t} + G K e^{\alpha t} \quad \text{and} \\ \alpha X_0 &= A X_0 + G K \quad \text{or} \\ (\alpha I - A) X_0 &= G K . \end{aligned}$$

Solving for X_0 we get

$$X_0 = (\alpha I - A)^{-1} G K . \quad (2.99)$$

The system matrix A can be reduced by similarity transformations to a diagonal matrix Λ . This is done by the EIGENP routine [25, 26]. The process is

$$A = S \Lambda S^{-1} \quad (2.100)$$

where

Λ is an $(N \times N)$ diagonal matrix composed of the complex system eigenvalues, and
 S is a complex $(N \times N)$ matrix whose columns are the system eigenvectors.
This matrix is referred to as the *modal matrix*.

The substitution of (2.100) into (2.99) leads to

$$\begin{aligned} X_0 &= (S \alpha S^{-1} - S \Lambda S^{-1})^{-1} G K \\ X_0 &= S (\alpha I - \Lambda)^{-1} S^{-1} G K \quad \text{or} \\ X_0 &= S (j \omega I - \Lambda)^{-1} S^{-1} G K(\omega) . \end{aligned} \quad (2.101)$$

Equation (2.101) represents the AC solution to the state variable equation given in (2.94).

Equation (2.95) represents the non-state variable quantities. The AC solutions for these voltages and currents are consistent with the transient equations given in subsection 2.3.1, with the understanding that their implementation requires complex arithmetic. The simpler equations below are substituted for four quantities:

$$\begin{aligned} I_1 &= j \omega C_{11} V_1 \\ I_4 &= j \omega C_{44} V_4 \\ V_3 &= j \omega (L_{36} I_6 + L_{33} I_3) \\ V_6 &= j \omega (L_{36}^T I_3 + L_{66} I_6) \end{aligned}$$

A and G Matrix calculation

SCEPTRE defines the state variables as the link inductor currents, I_3 , and the capacitor branch voltages, V_4 . SCEPTRE also permits both voltage sources, E_7 , and current sources, J_8 . This leads to the following definition for X and V .

$$X = \left[\begin{array}{c} I_3 \\ V_4 \end{array} \right] \Bigg\} N, \quad V = \left[\begin{array}{c} E_7 \\ J_8 \end{array} \right] \Bigg\} N_S.$$

Rewriting and partitioning Equation (2.94) in an alternative matrix form produces:

$$\left[\begin{array}{c} \dot{I}_3 \\ \dot{V}_4 \end{array} \right] = \left[\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right] \left[\begin{array}{c} I_3 \\ V_4 \end{array} \right] + \left[\begin{array}{cc} G_{11} & G_{12} \\ G_{21} & G_{22} \end{array} \right] \left[\begin{array}{c} E_7 \\ J_8 \end{array} \right],$$

as separate equations we have

$$\begin{aligned} \dot{I}_3 &= A_{11}I_3 + A_{12}V_4 + G_{11}E_7 + G_{12}J_8 \\ \dot{V}_4 &= A_{21}I_3 + A_{22}V_4 + G_{21}E_7 + G_{22}J_8. \end{aligned}$$

The transient equations of subsection 2.3.1 [25, 26] were used in the derivation of the necessary submatrices.

The submatrices $A_{11}, A_{12}, A_{21}, A_{22}$ for the case involving lumped, passive, time invariant R, L and C elements are given by Equations (2.102), (2.103), (2.104), and (2.105).

$$A_{11} = (-M_{LI}) * B_{35} * R_{55} * [B_{35}^T - B_{25}^T * (M_{RI}) * B_{25} * R_{55} * B_{35}^T] \quad (2.102)$$

$$A_{12} = (-M_{LI}) * [B_{34} - B_{35} * R_{55} * B_{25}^T * (M_{RI}) * B_{24}] \quad (2.103)$$

$$A_{21} = (C_{44}I) * [B_{14}^T * (M_{SI}) * B_{14} * S_{44} - I] * [B_{24}^T * (M_{RI}) * B_{25} * R_{55} * B_{35}^T - B_{34}^T] \quad (2.104)$$

$$A_{22} = (C_{44}I) * [B_{14}^T * (M_{SI}) * B_{14} * S_{44} - I] * [B_{24}^T * (M_{RI}) * B_{24}] \quad (2.105)$$

where

$$\begin{aligned} M_{LI} &= [L_{33} + L_{36} * B_{36}^T + B_{36} * L_{63} + B_{36} * L_{66} * B_{36}^T]^{-1} \\ M_{RI} &= [R_{22} + B_{25} * R_{55} * B_{25}^T]^{-1} \\ M_{SI} &= [S_{11} + B_{14} * S_{44} * B_{14}^T]^{-1}. \end{aligned}$$

The submatrices $G_{11}, G_{12}, G_{21}, G_{22}$, for the case involving independent voltage and/or current sources, are given by Equations (2.106) through (2.109).

$$G_{11} = (-M_{LI}) * [B_{37} - B_{35} * R_{55} * B_{25}^T * (M_{RI}) * B_{27}] \quad (2.106)$$

$$G_{12} = (-M_{LI}) * [B_{35} * R_{55} * (B_{85}^T - B_{25}^T * (M_{RI}) * B_{25} * R_{55} * B_{85}^T)] \quad (2.107)$$

$$G_{21} = (C_{44}I) * [(B_{14}^T * (M_{SI}) * B_{14} * S_{44} - I) * B_{24}^T * (M_{RI}) * B_{27}] \quad (2.108)$$

$$G_{22} = (C_{44}I) * [(B_{14}^T * (M_{SI}) * B_{14} * S_{44} - I) * (B_{24}^T * (M_{RI}) * B_{25} * R_{55} * B_{85}^T - B_{84}^T)] \quad (2.109)$$

2.5.2 Circuits Containing Linearly Dependent Sources

Reduction to Canonical Form

SCEPTRE permits the user to define voltage and current sources that are linear functions of resistor voltages and currents. These are type E_Y and J_X sources. Let u be a vector composed of these sources given by, $u^T = [E_Y J_X]$. Then a circuit containing these sources can be characterized by

$$\dot{X} = AX + GV + Hu \quad (2.110)$$

where H is an $(N \times N_D)$ matrix of real coefficients relating the dependent sources to the state variables. N_D is the number of linearly dependent sources.

The problem is to reduce Equation (2.110) to canonical form, namely

$$\dot{X} = (A_{\text{EXT}})X + (G_{\text{EXT}})V$$

where (A_{EXT}) is an $(N \times N)$ matrix of real coefficients describing the extended network. (G_{EXT}) is an $(N \times N_S)$ matrix of real coefficients relating the external sources to the extended network. Equation (2.110) may be rewritten in the following way:

$$\begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_Y \\ J_X \end{bmatrix}. \quad (2.111)$$

By definition,

$$E_Y = k_1 V_2 + k_2 V_5 \quad (2.112)$$

and

$$J_X = k_3 I_2 + k_4 I_5 \quad (2.113)$$

where k_1, k_2, k_3, k_4 are matrices relating each dependent source to the appropriate resistor voltage or current on which it depends.

Since $V_2 = R_{22}I_2$ and $I_5 = G_{55}V_5$ substitution into (2.112) and (2.113) yields:

$$\begin{bmatrix} E_Y \\ J_X \end{bmatrix} = \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix}. \quad (2.114)$$

Substituting (2.114) into (2.111) gives

$$\begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} \begin{bmatrix} I_2 \\ V_5 \end{bmatrix}. \quad (2.115)$$

The vector $[Y]^T = [I_2 V_5]$ can be expressed in term of state variables and independent sources by observing that Y' is a subset of the non-state variable outputs Y , where $Y = CX + DV$. That is

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = Y' = C'X + D'V. \quad (2.116)$$

However, subsection 2.3.4 gives

$$\begin{bmatrix} I_2 \\ V_5 \end{bmatrix} = M_{\text{RG}}^{-1} \begin{bmatrix} -B_{24}V_4 & -B_{27}E_7 \\ B_{35}^T I_3 & +B_{85}^T J_8 \end{bmatrix}$$

where

$$M_{\text{RG}} = \begin{bmatrix} (R_{22} + B_{2Y}k_1 R + 22) & (B_{25} + B_{2Y}k_2) \\ (-B_{25}^T - B_{X5}^T k_3) & (G_{55} - B_{X5}^T k_4 G_{55}) \end{bmatrix}$$

therefore

$$C' = M_{\text{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \quad (2.117)$$

and

$$D' = M_{\text{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix}. \quad (2.118)$$

Using (2.118), (2.117) and (2.116) we can write Equation (2.115) in terms of state variables and independent sources

$$\begin{aligned} \begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} \\ &+ \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} M_{\text{RG}}^{-1} \left(\begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} + \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} \right). \end{aligned}$$

Collecting terms we have

$$\begin{aligned} \begin{bmatrix} \dot{I}_3 \\ \dot{V}_4 \end{bmatrix} &= \left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} M_{\text{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \right) \begin{bmatrix} I_3 \\ V_4 \end{bmatrix} \\ &+ \left(\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} M_{\text{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \right) \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} \end{aligned} \quad (2.119)$$

where Equation (2.119) represents a reduction of Equation (2.110) to a canonical form as required with

$$A_{\text{EXT}} = \left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} M_{\text{RG}}^{-1} \begin{bmatrix} 0 & -B_{24} \\ B_{35}^T & 0 \end{bmatrix} \right)$$

and

$$G_{\text{EXT}} = \left(\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} k_1 R_{22} & k_2 \\ k_3 & k_4 G_{55} \end{bmatrix} M_{\text{RG}}^{-1} \begin{bmatrix} -B_{27} & 0 \\ 0 & B_{85}^T \end{bmatrix} \right).$$

H Matrix Calculation

Equation (2.119) is complete except for a description of the H matrix which is necessary for computational purposes.

It is easily shown that the H matrix is quite similar in structure to the G matrix by observing the equivalence in the following equation

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \end{bmatrix} + \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_Y \\ J_X \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & H_{11} & H_{12} \\ G_{21} & G_{22} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} E_7 \\ J_8 \\ E_Y \\ J_X \end{bmatrix}.$$

This indicates that the G and H submatrices differ only by post-multiplications by terms which relate the circuit connectivity to the appropriate sources.

Thus

$$G = \left[\begin{array}{c|c} (g_1)B_{37} + (g_2)B_{27} & (g_4)B_{85}^T \\ \hline & +(g_3)B_{27} & (g_5)B_{85}^T + (g_6)B_{84}^T \end{array} \right]$$

and

$$H = \left[\begin{array}{c|c} (g_1)B_{3Y} + (g_2)B_{2Y} & (g_4)B_{X5}^T \\ \hline & +(g_3)B_{2Y} & (g_5)B_{X5}^T + (g_6)B_{X4}^T \end{array} \right]$$

where

$$\begin{aligned} g_1 &= -M_{LI} \\ g_2 &= M_{LI}(B_{35}R_{55})(B_{25}^T M_{RI}) \\ g_3 &= C_{44}I(B_{14}^T M_{SI}B_{14}S_{44} - I)B_{24}^T M_{RI} \\ g_4 &= -M_{LI}(B_{35}R_{55}[I - (B_{25}^T M_{RI})(B_{25}R_{55})]) \\ g_5 &= C_{44}I(B_{14}^T M_{SI}B_{14}S_{44} - I)(B_{24}^T M_{RI}B_{25}R_{55}) \\ g_6 &= -C_{44}I(B_{14}^T M_{SI}B_{14}S_{44} - I) . \end{aligned}$$

2.5.3 Circuits requiring Time Derivatives of Independent Sources

Derivation of the AC Solution

If the circuit topology is such that a source time derivative is required in order to perform a transient analysis⁸, then this same requirement is also imposed on an AC analysis. However, there is one important difference. In the AC analysis program, unlike the transient program, the user is not required to supply the derivative. The AC program automatically supplies the needed information and proceeds.

The derivatives are treated internally as follows: let this situation be characterized by

$$\dot{X} = AX + GV + Q\dot{V} \quad \text{where} \quad V_k(t, \omega) = K_k(\omega) e^{j\omega t} . \quad (2.120)$$

Then the solution of Equation (2.120) is of the form

$$X = X_0 e^{j\omega t} .$$

Thus

$$\begin{aligned} \frac{d}{dt}(X_0 e^{j\omega t}) &= A(X_0 e^{j\omega t}) + G(K e^{j\omega t}) + Q \frac{d}{dt}(K e^{j\omega t}) \\ j\omega X_0 e^{j\omega t} &= AX_0 e^{j\omega t} + GK e^{j\omega t} + j\omega QK e^{j\omega t} \end{aligned}$$

and

$$\begin{aligned} (j\omega I - A)X_0 &= GK + j\omega QK \\ X_0 &= (j\omega I - A)^{-1}(GK + j\omega QK) . \end{aligned}$$

⁸Independent voltage sources together with a capacitor tie-set, or independent current sources together with an inductor cut-set

Substituting our similarity transformation gives

$$\begin{aligned} X_0 &= S(j\omega I - A)^{-1}S^{-1}(GK + j\omega QK) \\ X_0 &= S(j\omega I - A)^{-1}[(S^{-1}GK) + j\omega(S^{-1}QK)] . \end{aligned} \quad (2.121)$$

Equation (2.121) represents the AC solution to the state variable Equation given in (2.120). The second term in the brackets represents the additional computation required, at each frequency, due to the presence of all source derivatives.

Q Matrix Calculation

This matrix was also derived, using the differential equations of Section 2.3.1 It is listed here for continuity.

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

where

$$\begin{aligned} Q_{11} &= 0 \\ Q_{12} &= (-M_{LI})[B_{36}L_{66}B_{86}^T + L_{36}B_{86}^T] \\ Q_{21} &= (-C_{44}I)B_{14}^T M_{SI}B_{17} \\ Q_{22} &= 0 . \end{aligned}$$

2.5.4 Circuits Containing Linearly Dependent Sources and Requiring Time Derivatives of Independent Sources

This case presents not difficulties. It is simply a combination of case 2 and 3 (see page 35).

3 System Operation

3.1 Introduction

The SCEPTRE System is a FORTRAN IV computer program written for the IBM 7090/95 and 360 Data Processing Systems. It consists of two major phases. The first, called the Program Generator, creates (on a disk or tape data file¹) another FORTRAN IV computer program containing circuit equations for electrical networks. SCEPTRE does this automatically, using the input data describing the circuit to be analyzed. The second phase, the Circuit Solution Executive, computes the circuit response by solving these equations generated in the FORTRAN IV program.

Operation of the SCEPTRE System depends upon the Monitor System, IBSYS (operating system OS/360) which controls its execution. Under control of IBSYS (OS/360), SCEPTRE executes in two phases as separate job steps that are loaded and executed sequentially. Prior to loading, however, IBSYS (OS/360) performs any required FORTRAN compilation. Programs to be compiled do not have to reside on the standard input file upon which the input data are stored. Instead, the system can be instructed, via the job control language, JCL to compile the load programs from an alternate input file. SCEPTRE uses this feature of IBSYS (OS/360) as a means of linking its two job steps. This is illustrated in the System Flow Diagram, Figure 3.1.

3.2 Program Generator

The Program Generator is an executive program that controls the inputting of circuit description data, generation of a FORTRAN IV Subprogram for calculating circuit response, generation of circuit parameter data, storage of circuit models on library files, restart of discontinued runs, and re-outputting of computed results. Each of these six program tasks as well as the Program Generator (EXEC1) are described in the flow diagrams, figures 3.2 through 3.7.

3.2.1 Circuit Description Processor

The SCEPTRE circuit description language is used to describe electrical networks. This application-oriented language is powerful, easy to learn and use, and nearly format free. With it, circuits composed of fundamental circuit elements and prestored circuit models can be described. The types of fundamental electrical characteristics allowed are resistance, capacitance, inductance, current and voltage sources, and mutual inductance. Using the same components and circuit description language, equivalent circuit models of devices such as transistors and diodes can be described and stored on a library file for future use. When a stored model is referenced in a circuit description, it is located on the specified library file and its elements appropriately substituted into the circuit being described. Often, when a model is called out in describing a circuit, the desired parameter values are different from those originally specified. Rather than permanently change the stored model, the

¹This term "file" in this section refers to disk or tape file for the S/360 and to a tape file for the 7090/94.

SCEPTRE circuit description language allows model parameter values to be changed when model element substitution takes place.

3.2.2 Model Editor

Using the SCEPTRE circuit description language, circuit models constructed by the user can be stored on a model library file by the Model Editor program. One permanent library file and one temporary library file are provided for storing models. Circuit models may be an arbitrary n-terminal configuration of the allowed fundamental circuit elements. Any model so described can be stored on either library file using the Model Editor Program.

3.2.3 Circuit Equation Generator

After the description of the circuit to be analyzed has been reconstructed in memory of the Circuit Description Processor, the FORTRAN IV sub-program called **SIMUL8** is created. It contains D-C steady-state, transient, and/or AC solution equations, depending on the type of analysis requested by the user. These equations are based on the formulation presented in Section 2 of this report. The **SIMUL8** program is written on an output file (PROGRAM SAVE TAPE) and stored until the second phase of SCEPTRE operation, when it will be compiled and executed.

3.2.4 Data Generator and Rerun Processor

Essentially the **SIMUL8** program contains only the circuit equations for the network under investigation. The parameter or component values of the network are stored separately as input data to the **SIMUL8** program. The Data Generator program organizes and stores the circuit data on the PROGRAM SAVE TAPE.

Once a circuit has been described to SCEPTRE, multiple or repeated circuit solutions can be run by changing parameter values between runs. The circuit description language is used to specify the number of repeated runs, each known as a rerun, and the changes in parameter values desired for each rerun. The Rerun Processor interprets and processes this information. The Data Generator then creates and stores one block of circuit data for each rerun on the PROGRAM SAVE TAPE. Since only parameter values are changed between reruns (i.e., no topological changes), the **SIMUL8** program containing the original circuit equations is simply re-executed during the solution phase using the data created for each rerun.

3.2.5 Continue Processor

The Continue Processor allows previously terminated transient solution runs to be restarted and continued. In addition, it is necessary that the PROGRAM SAVE TAPE from the original transient solution run be available. This file contains, in addition to the **SIMUL8** program, all of the parameter values and data required to continue the transient solution. The Continue Processor also allows certain run control parameters to be changed and used for the continuation of the original transient run.

3.2.6 Re-Output Processor

The SCEPTRE user may wish to re-create both the printed and plotted outputs for a particular transient analysis run. This can be done, providing the OUTPUT SAVE TAPE is saved from the original solution run, by using the Re-Output Processor. Several changes in the output can be made during a Re-Output run. For example:

- Output labels can be changed
- New quantities can be plotted
- The order in which output quantities are printed out and plotted can be changed
- Printing and plotting of output quantities can be suppressed.

3.3 Solution Executive

In the second phase of SCEPTRE operation, the FORTRAN IV sub-program **SIMUL8** is compiled and executed. During this job step, IBSYS (OS/360), is instructed to read the **SIMUL8** program from the PROGRAM SAVE TAPE, compile it, and then link edit it along with the other required programs.

Execution commences, as shown in figure 3.7 with the reading of the run control parameters from the PROGRAM SAVE TAPE. Then the **SIMUL8** program is called. It reads the remaining circuit parameter values stored on the PROGRAM SAVE TAPE and then enters the circuit solution equations. The circuit equations are solved, thus computing the state-variable derivatives. Calling the selected numerical integration routine then produces new values of the state variables for the next time step. At the conclusion of each successful integration step, the requested output quantities are buffered in memory. When the buffer is full, it is written on the OUTPUT SAVE TAPE as binary data. After the circuit solution is complete, control is returned to the Solution Executive Program, whereupon the contents of the OUTPUT SAVE TAPE (computed results) are re-formatted in lists and graphs and stored on the SYSTEM OUTPUT TAPE for peripheral processing.

If any reruns were requested, control is then returned to **SIMUL8** for re-execution of the solution phase and subsequent outputting. This recycling is continued until all circuit reruns have been processed.

At the conclusion of each transient solution, the critical parameter values and data are stored on the PROGRAM SAVE TAPE. Thus, if the file is saved at the end of the run, the solution can be continued at some future time. Use of the Continue Processor allows previously terminated solution runs to be restarted and continued with changes in the run control parameters.

Transient solution runs may be terminated or simply saved by the computer operator by depressing sense switch No. 6 on the 7090/94 console, or by appropriate changes to the Data Definition (DD) cards on the S/360². Using this feature, a PROGRAM SAVE TAPE could, for example, be generated every 15 minutes on long running transient solutions, eliminating the need for repeating previous calculations in the event of an abnormal run termination.

If, at the conclusion of a transient analysis run, the OUTPUT SAVE TAPE is saved, the Re-Output Processor may be used to reproduce lists and graphs of any of the circuit quantities originally requested for output. In addition, graphs of variables plotted against variables other than time which may not have been requested originally can be conveniently produced.

²See subsection 7.1.3.2 of Volume 1

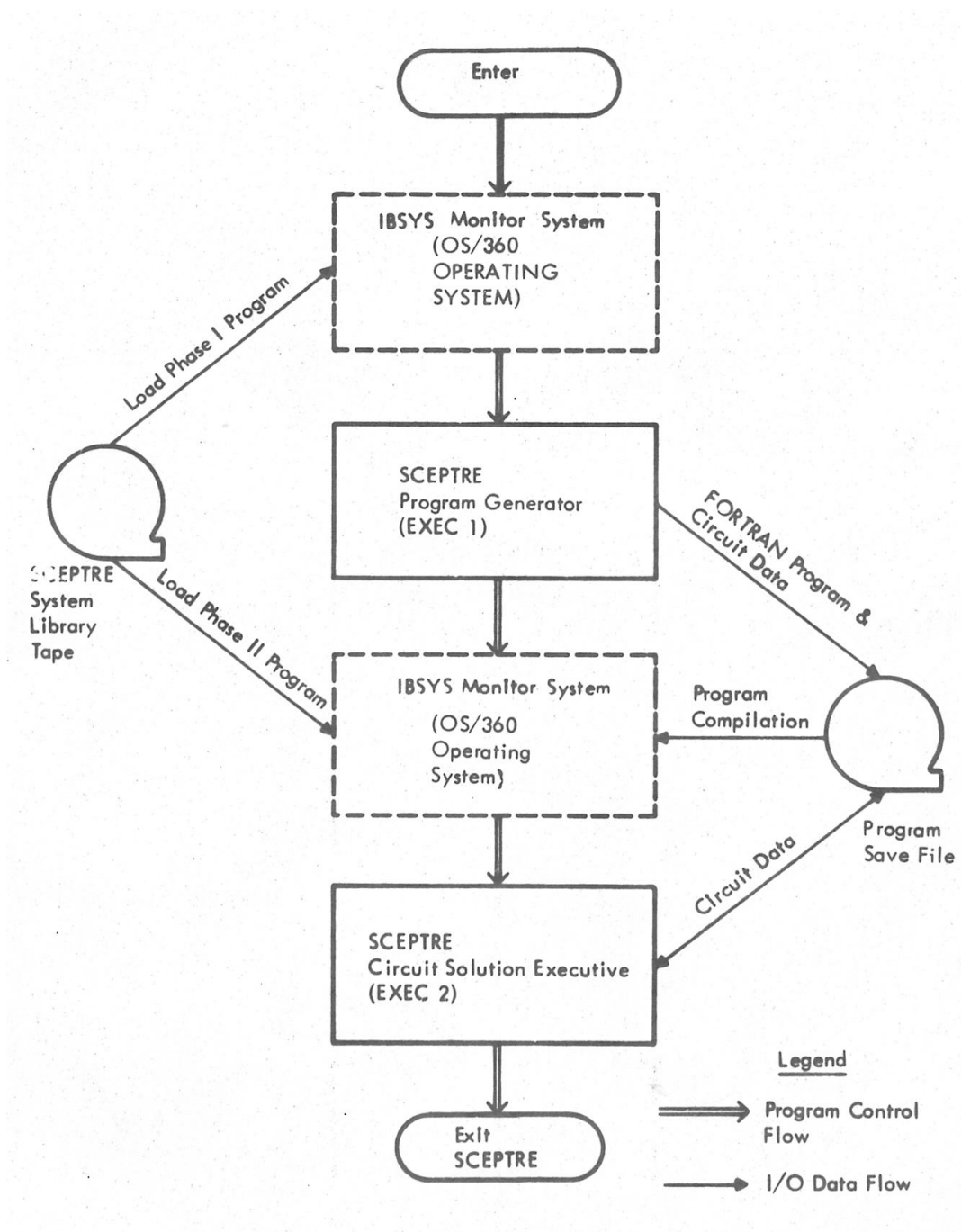


Figure 3.1: System Flow Diagram

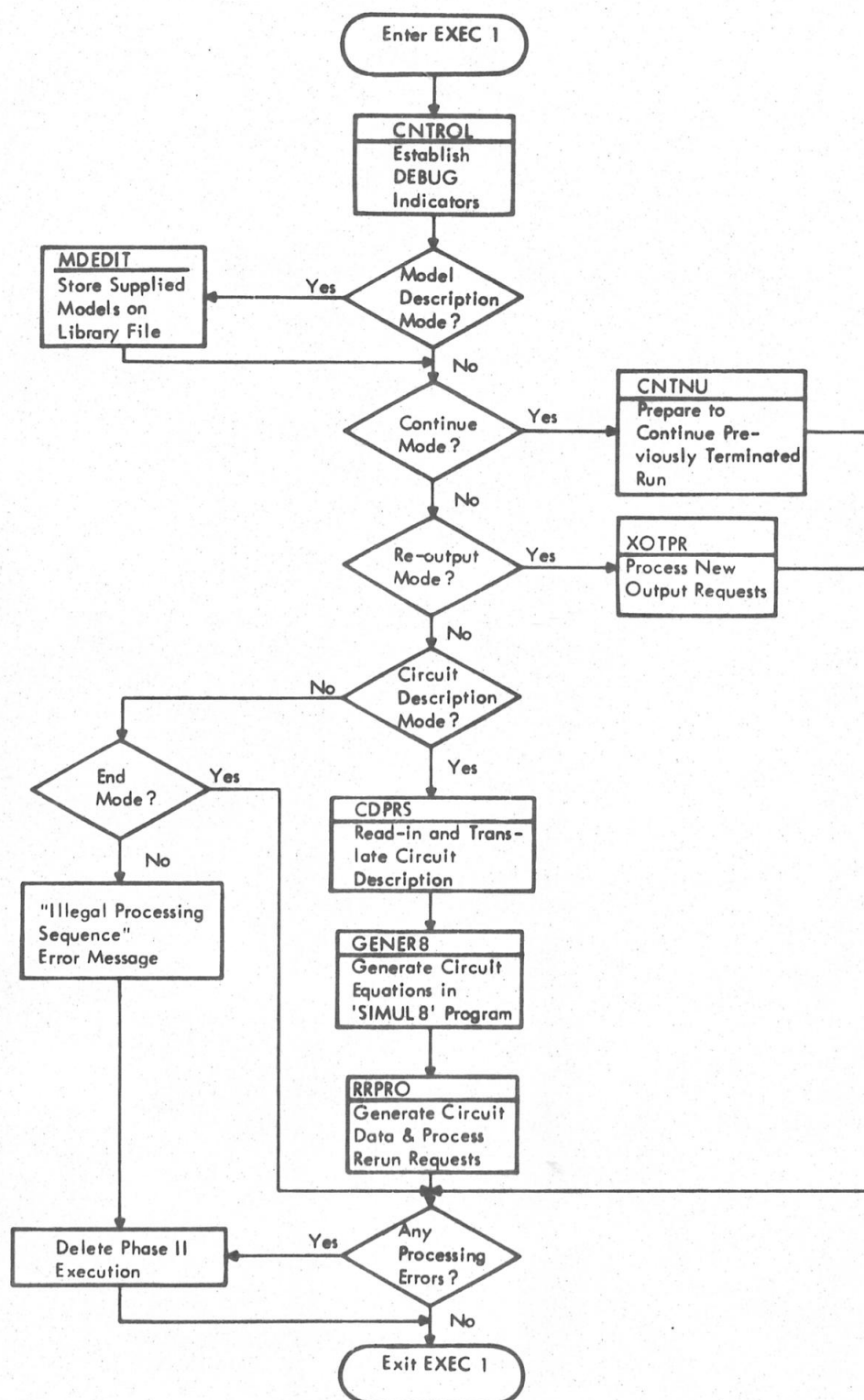


Figure 3.2: Program Generator (EXEC1) Flow Diagram

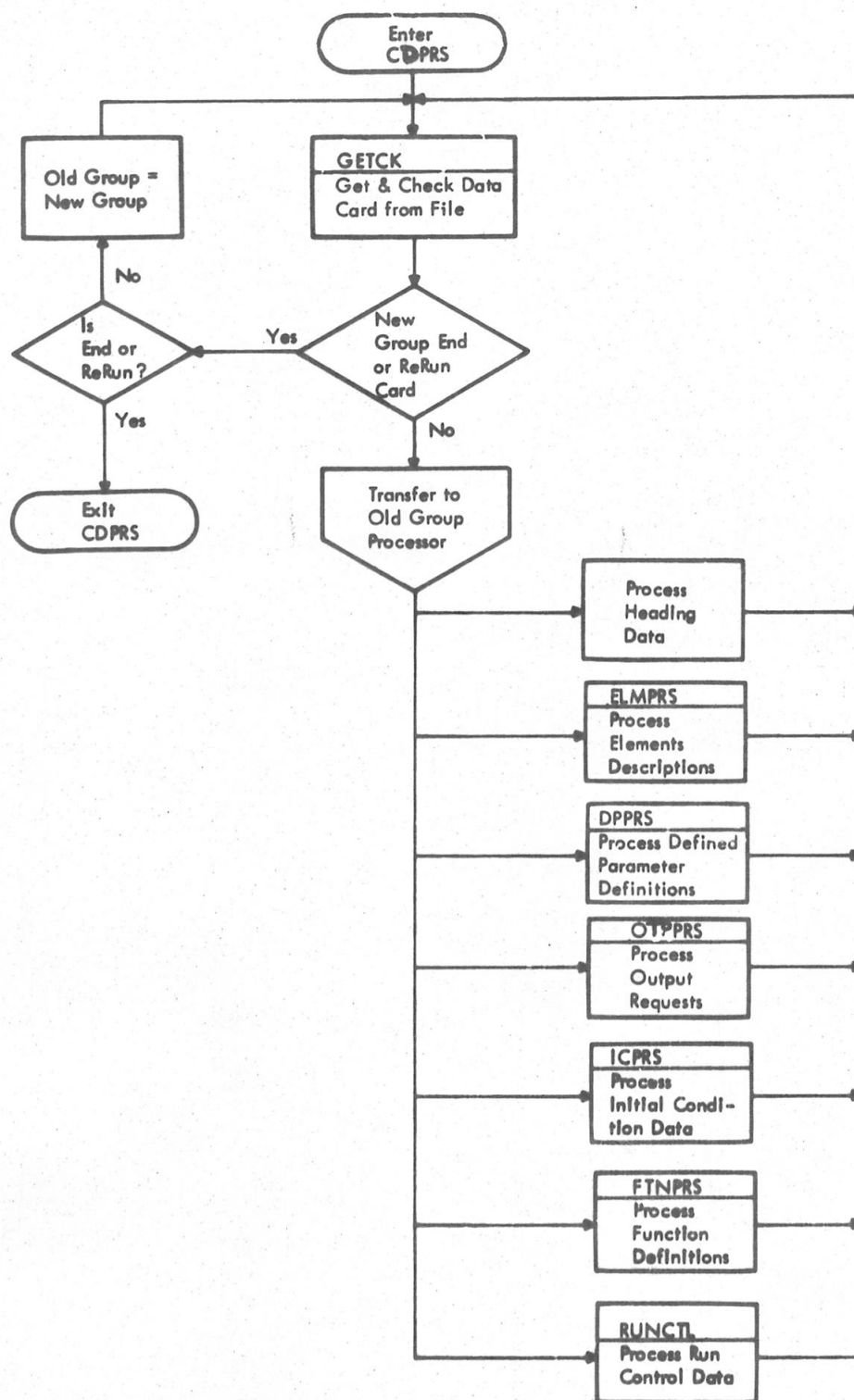


Figure 3.3: Circuit Description Processor (CDPRS) Flow Diagram

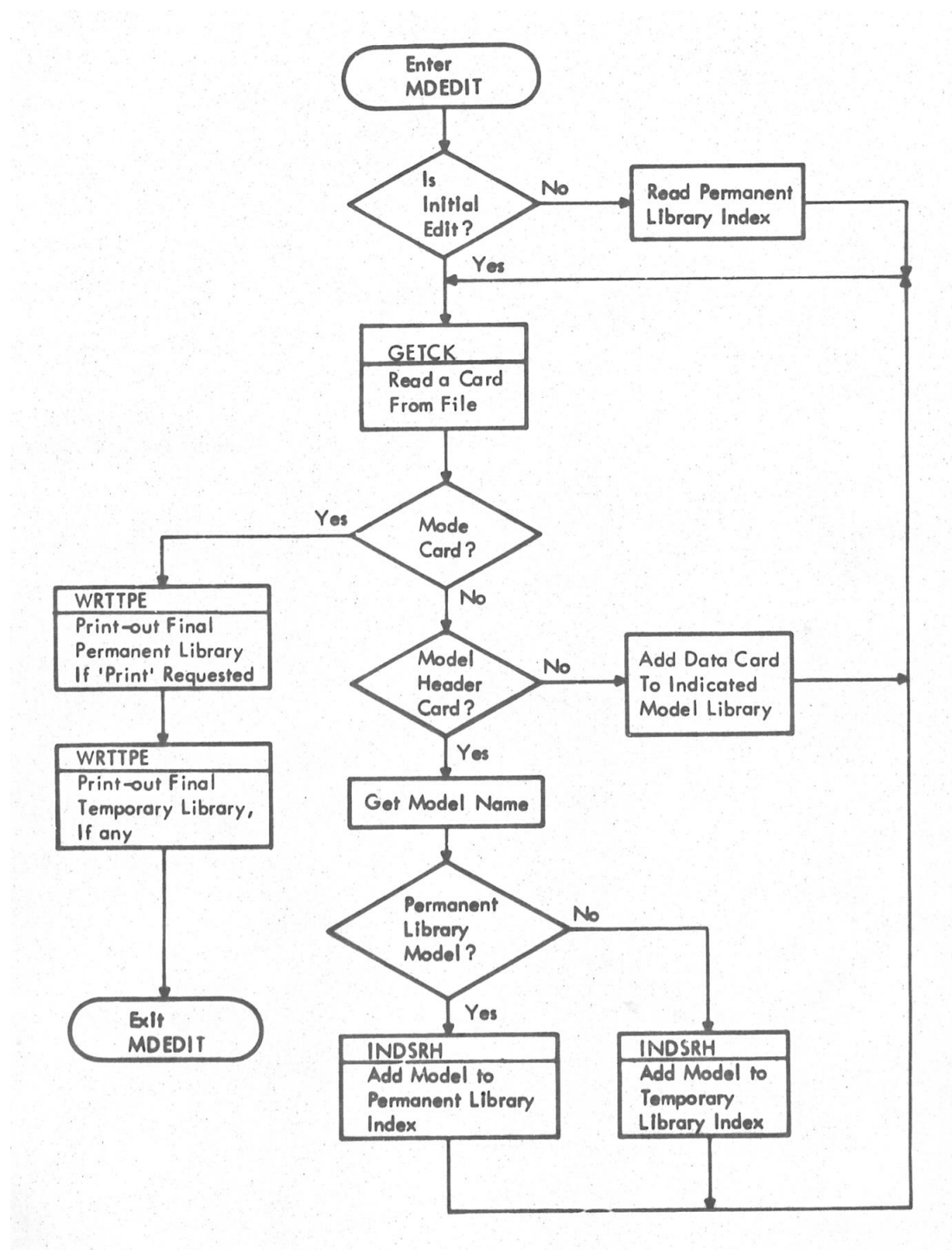


Figure 3.4: Model Editor (MDEDIT) Flow Diagram

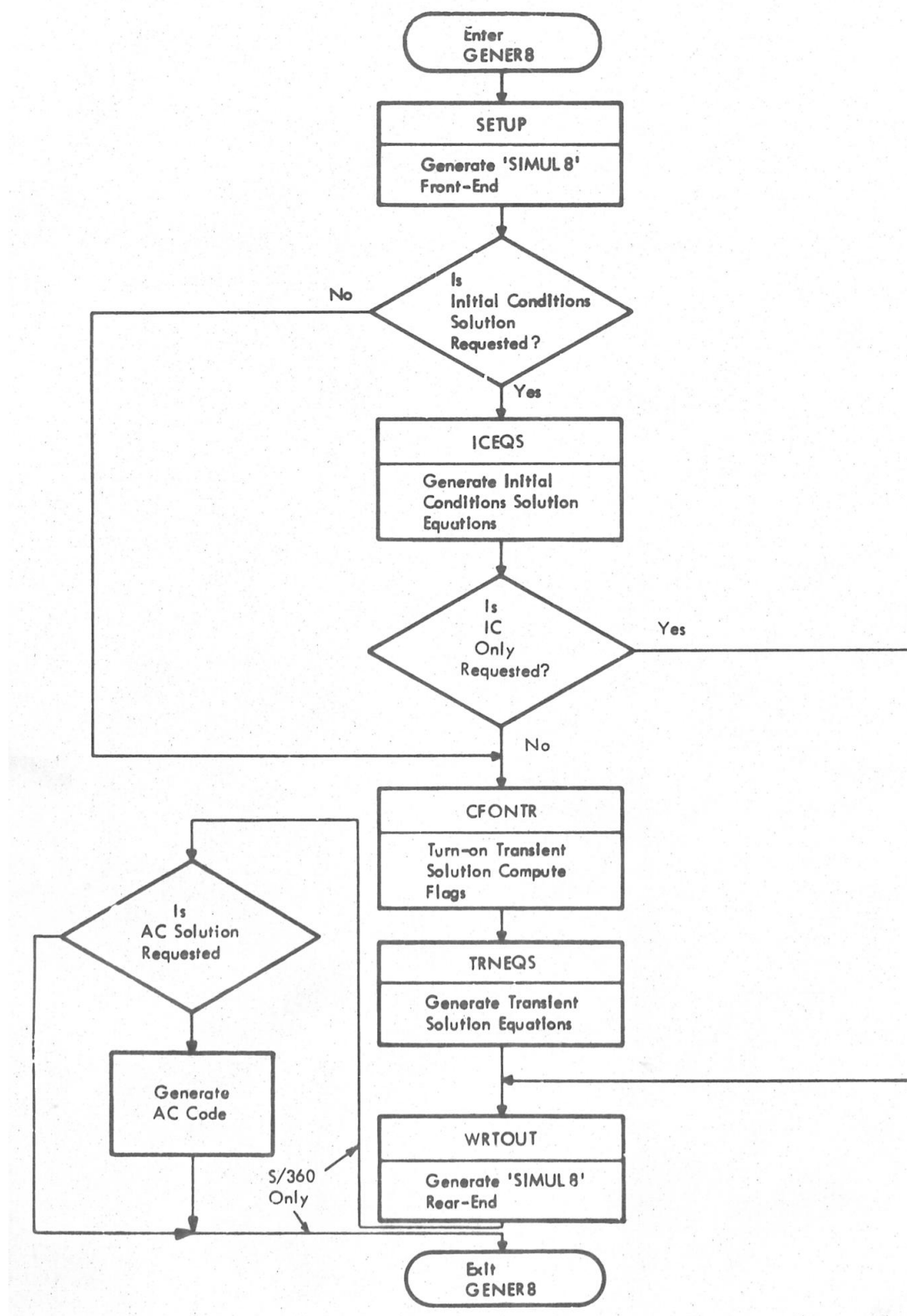


Figure 3.5: Circuit Equation Generator (GENER8) Flow Diagram

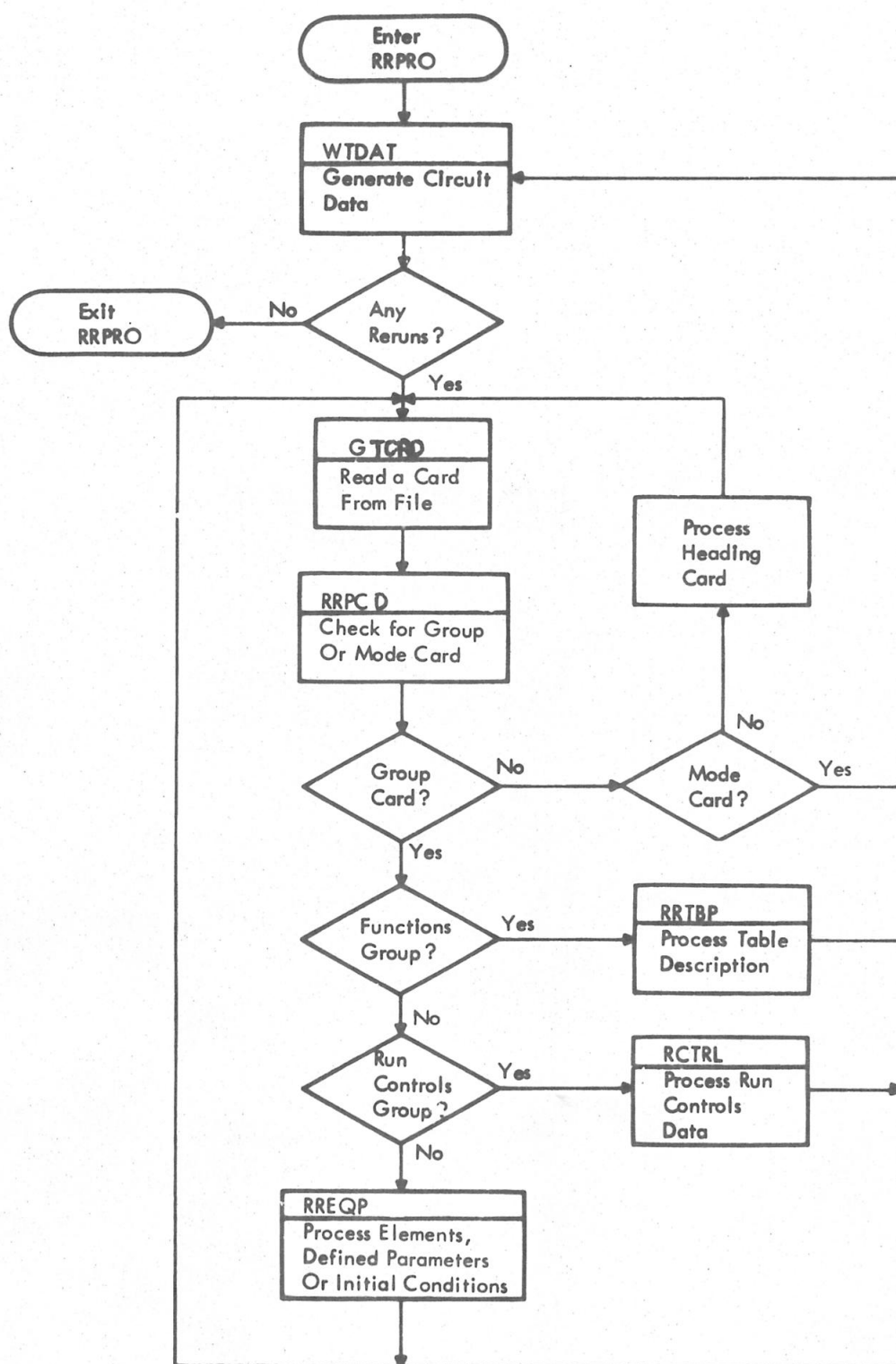


Figure 3.6: Rerun Processor and Data Generator (RRPRO) Flow Diagram

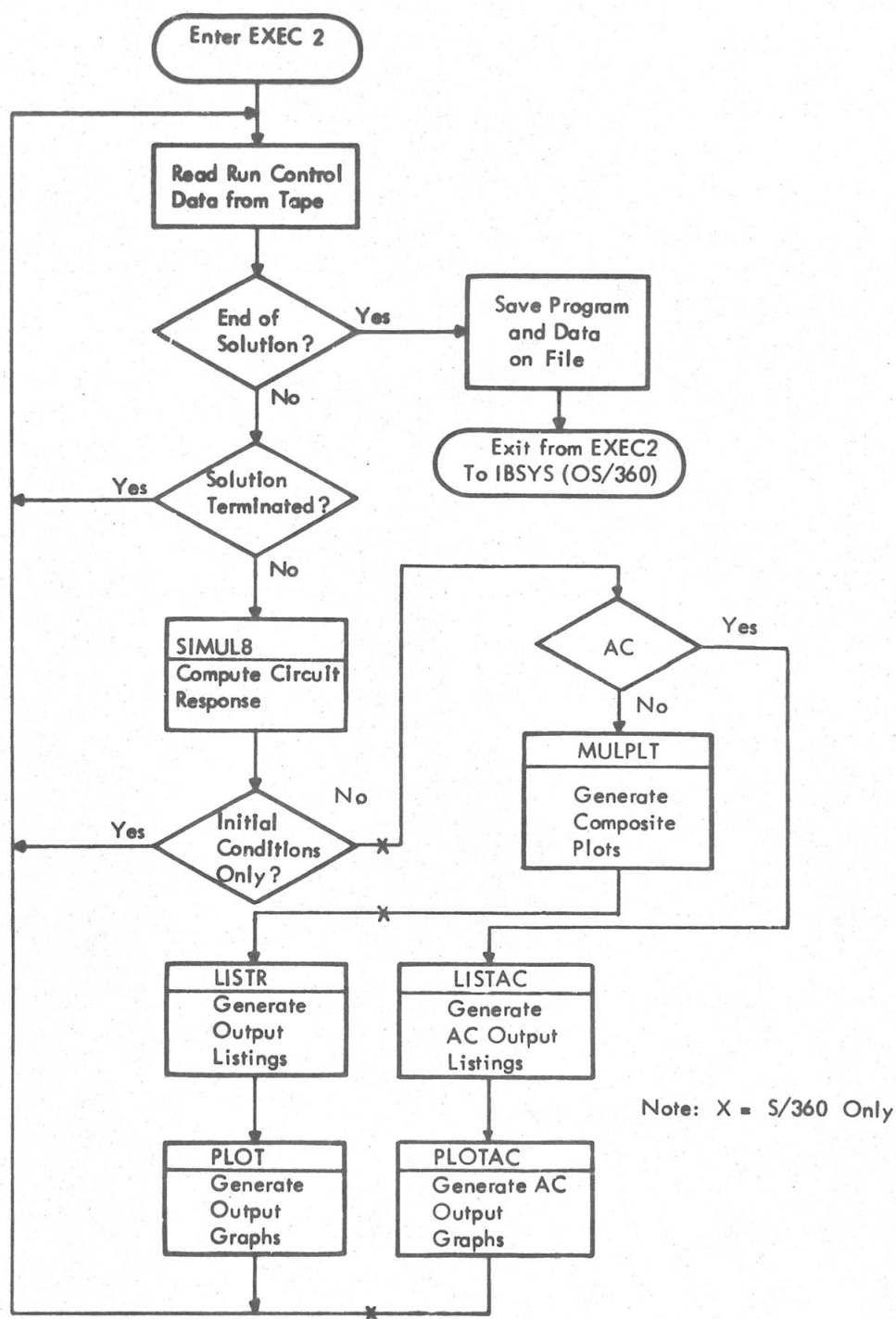


Figure 3.7: Circuit Solution Executive (EXEC2) Flow Diagram

4 Sparse Matrix Techniques

A very common operation that is performed in many aspects of scientific digital computation is solution of the system

$$A X = b \quad (4.1)$$

where A is a coefficient matrix, b is a known vector of forcing functions and X is the vector of unknown quantities to be determined. Conventional techniques have of course long been known. Gaussian elimination or Gauss-Jordan methods are standard in most numerical analysis texts. The need for improvement becomes evident however when A becomes large. If A is of order n , these methods require a little more than n^2 words of core storage. When $n = 100$, more than 10,000 words of core are committed to this operation alone. Yet the larger A is, it is usually true that the percentage of zero elements, or sparseness, increases. A sparseness of 90 percent is not unusual when $n > 100$. A sparse matrix technique then, can reasonably be defined as any method that takes advantage of the inherent sparseness of an often used matrix to perform computation with some significant saving of storage and sometimes even solution time.

The most pressing need for some type of sparse matrix technique in SCEPTRE occurs in the DC algorithm. The nature of the mathematical formulation is such that the A matrix corresponding to Equation (4.1) is equal in size to the number of network resistors plus semiconductor devices. Experience has shown that the largest A matrix that could be solved within 200 kBytes in the System/360 was about 120×120 . This problem caused some runs to fail that otherwise could have been easily accommodated.

The specific method that was implemented is based upon triangular decomposition [25]. If A^{-1} exists, then A can be decomposed as

$$A = L U \quad (4.2)$$

where L is a lower triangular matrix and U is an upper triangular matrix as shown below:

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \quad U = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ 0 & 0 & 1 & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

The matrices L and U can be generated from the original A matrix by use of the recursive relations

$$l_{iq} = a_{iq} - \sum_{k=1}^{q-1} l_{ik} u_{kq} \quad \text{with } l_{iq} = 0 \text{ for } i < q$$

$$u_{pj} = \frac{a_{pj} - \sum_{k=1}^{p-1} l_{pk} u_{kj}}{l_{pp}} \quad \text{with } u_{pj} \begin{cases} = 0 & \text{for } p > j \\ = 1 & \text{for } p = j \end{cases}$$

Once L and U have been generated, solution may proceed in a straight forward manner. Since the basic problem can be written in the form of Equation (4.1) where vector X represents the unknowns, then substitution of Equation (4.2) in (4.1) gives

$$L U X = b . \quad (4.3)$$

Define

$$Y = U X \quad (4.4)$$

which transforms Equation (4.3) to

$$L Y = b . \quad (4.5)$$

Equation (4.5) is in a form suitable to solve for Y by a simple front substitution process. Once Y is known, Equation (4.4) can be solved by back substitution to arrive at the desired system unknown X . The attractiveness of this method is that it lends itself to an indexing scheme that can work with only nonzero elements of L and non-zero off diagonal elements of U . If A is large and sparse, a significant storage advantage can be realized.

The new sparse technique was tested on two circuits that were far larger than could have been handled with the existing method within 200 kBytes of core. The first circuit was composed of 200 resistors and semiconductor junctions and the resulting A matrix was well over 90 percent sparse. Accurate convergence was obtained in seven passes. The second circuit contained 290 resistors and semiconductor junctions and a check of the A matrix revealed over 95 percent sparseness. This circuit also converged accurately in seven passes. If this same network was solved with the conventional technique, approximately 800 kBytes of core storage would have been required. A series of smaller networks were also solved with the sparse technique and in every case the solution converged to the same result in the same number of passes as did the existing method.

Since this method is attractive for large A only, it has been programed into SCEPTRE such that it becomes operational for networks containing 70 or more elements with DC analysis requested. The conventional Gauss-Jordan technique will continue to be used for smaller networks. The sparse technique will also be applied to the transient portion of the program for the solution of larger problems. It is clear that this technique is a necessary prerequisite in eventually permitting the analysis of very large circuits.

5 Assembler Language Input/Output (I/O)

SCEPTRE was originally written in FORTRAN IV on the 7094 Computing System and was later converted to OS/360 FORTRAN IV (H-level, Optimization =2). The decision to program in FORTRAN facilitated conversion of the program for use on other computing systems; but it also resulted in increased CPU time requirements for the setup phase. In an effort to reduce this time requirement it was determined that FORTRAN I/O accounted for more than 50 percent of the total CPU time used, particularly for small problems. Since the CPU requirements for assembler language I/O is at least 80 % less than that required by FORTRAN I/O with `FORMAT`, the FORTRAN I/O was replaced with assembler language I/O with the exception of output directed to the printer. The implementation of assembler language I/O into the setup phase and the resulting improvements are discussed in the following paragraphs.

Three assembler language subroutines were written to perform the functions of the FORTRAN I/O statements `READ`, `WRITE`, and `ENDFILE`. These subroutines utilize `GET` and `PUT` assembler language instructions to perform the I/O. Since these subroutines transfer data to and from core storage in blocks of 80 bytes (characters), the logical record lengths and the record formats for the affected data sets must be 80 bytes and fixed block (FB), respectively. The physical record lengths (i.e., `BLKSIZES`) can be any multiple of 80. The technique of selecting the proper data set for transferral of data is via the `DDNAME` of the data set. An eight character `DDNAME` for each data set is initialized in the `BLOCK DATA` subroutine. The `DDNAMEs` used are as shown in the SCEPTRE Program Control Deck in Volume I, Figure ???. If for some reason the user desires different `DDNAMEs`, he needs only make the appropriate changes in `BLOCK DATA` and his SCEPTRE Program Control Deck. No reformatting of the I/O is performed by the assembler language subroutines unless the amount of data to be transmitted is less than 80 bytes. In this case, the remainder of the record is padded with blanks.

The implementation of assembler language I/O did not require an extensive programming effort to reformat the I/O data. Much of the I/O of the setup phase involves reading and writing data where the core storage and data set representation are both character format. Conversion to a character format and vice-versa was necessary for the data where the core storage representation was integer or real constants. This includes circuit data stored in the program save data set and indices in the model library data sets. The reformatting of this data was performed by FORTRAN subroutines.

An effort was made to ensure that the addition of the assembler language I/O would not complicate the task of converting the SCEPTRE/360 System to other computers. All replaced FORTRAN I/O statements and their associated `FORMAT` were made comments and left in the program. To assist in the identification of these statements, the character 'X' was also added in column two of the commented statements. Most of the I/O conversion effort only requires that the calls to the assembler language routines and the characters "CX" from column 1 – 2 of the FORTRAN I/O statements be removed. Other statements to be deleted can be easily identified.

The results of the improvement in the execution speed of the setup phase are shown in table 5.1. The improvement was 60 to 75 percent for small problems. Although the improvement was only 30 percent for the large problem there was a significant reduction in total CPU time. As shown in table 5.1 the timing tests were runs made on an IBM System 360 Model 65 computer. The finished

product will appear transparent to the user in that the I/O data itself will retain its usual format.

Table 5.1: Timing Tests of the Setup Phase using FORTRAN I/O and Assembler Language I/O

Test Circuit	CPU Time Requirements for the Setup Phase (seconds)	
	FORTRAN I/O	Assembler Language I/O
Example 1 from SCEPTRE User's Manual	11	5
Example 2 from SCEPTRE User's Manual	24	9
Example 3 from SCEPTRE User's Manual	23	6
Example 4 from SCEPTRE User's Manual	11	5
221 Element Circuit (Transient Solution Only)	90	62

A Implicit Integration Implementation

A.1 Introduction

Two basic factors are involved in the amount of solution time required to complete any transient problem: the number of steps or solution increments needed and the amount of computation required per step. The first of these factors is controlled entirely by the type of numerical integration method that is used to integrate the system of differential equations generated. This system can generally be expressed by the matrix equation:

$$\dot{Y} = A Y + B U, \quad (\text{A.1})$$

where Y is the vector of state variables, $\dot{Y}$ is the vector of state variable derivatives and both A and B are appropriate coefficient matrices.

It is well known that the step sizes that can be taken by all explicit numerical integration methods are limited by the largest system eigenvalue. This limit is often referred to as the stability radius, and it differs from method to method. For example, if $|\lambda_m|$ is the absolute value of the largest eigenvalue of A in Equation (A.1), then for most well known explicit methods

$$hm \leq \frac{X}{|\lambda_m|} \quad (\text{A.2})$$

where hm is the maximum step size that may be taken. If larger steps are taken, the solution will begin to oscillate. The quantity X in Equation (A.2) is equal to 1 for Euler's method, 2 for the explicit trapezoidal method, 2.78 for Runge Kutta and 6 for TRAP 2 which is currently an optional choice in SCEPTRE. No constant value for X can be given for XPO, but it too has a stability limit. If $|\lambda_m|$ is large for any network, Equation (A.2) will force a small step size and therefore require that many solution steps be taken with an attendant rise in computer solution time.

There has been a recent trend in mathematical literature toward integration methods that do not exhibit the limitation inherent in Equation (A.2). A generalized form of these methods can be given as

$$Y_{n+1} = \sum_{i=0}^P a_i Y_{n-i} + h \sum_{i=1}^P b_i \dot{Y}_{n-i}, \quad (\text{A.3})$$

where the Y and h quantities remain as previously defined and the a and b quantities are usually, but not necessarily, constant. Multistep methods are introduced if index $P \neq 0$. The simplest derivative form of Equation (A.3) occurs if coefficients $a_0 = 1, b_{-1} = 1$ and all others are set equal to zero. We then have

$$Y_{n+1} = Y_n + h \dot{Y}_{n+1} \quad (\text{A.4})$$

which is commonly referred to as the *implicit* or *backward Euler technique*. This technique can be shown to be unconditionally stable, and has given marked speed advantages over all of the explicit

methods in SCEPTRE whenever the latter has been affected by the limitation implied in Equation (A.2). A major disadvantage of this particular method lies in its inaccuracy for some types of problems. Implicit or not, it is still a first order method and tests have indicated that both XPO and RUK can be considerably more accurate. The next well known variant of Equation (A.3) is set up if coefficients $a_0 = 1, b_{-1} = 0.5$ and all others are set to zero. This choice sets up

$$Y_{n+1} = Y_n + \frac{h}{2} \dot{Y}_n + \frac{h}{2} Y_{n+1} , \quad (\text{A.5})$$

a method called *implicit trapezoidal integration*. This method is considerably more accurate than the first order variety, but has lost the quality of unconditional stability. Under some computation circumstances oscillations can occur and render the analysis suspect. Therefore, after considerable testing and consultation with the Air Force Weapons Laboratory, it was decided to implement a multistep [17, 18] version of Equation (A.3) that automatically chooses $1 \leq P \leq 6$. A good deal of testing has been performed on this implementation and the resultant conclusions are given in the following paragraphs.

A.2 Manual Example 1

The first practical network to be run on SCEPTRE with implicit integration was example 1 in the SCEPTRE manual Vol. 1, Section 4.1. This particular network contains a significant spread in eigenvalues due to the small transistor capacitors and the larger load capacitor. A master run with two associated reruns was made and a summary of the number of integration steps and passes required by both the original explicit method and the new implicit method is given in Table A.1. Since this network did exhibit a significant spread in eigenvalues, the implicit results were much faster. No practical difference in accuracy was noted in any part of the runs.

Table A.1: Comparison of Explicit and Implicit Integration on Example 1

	Explicit		Implicit	
	Passes	Steps	Passes	Steps
Master	6441	3049	291	126
First Rerun	6065	2867	480	155
Second Rerun	5972	2832	255	112

Note: Large Eigenvalues can inhibit the step size and small Eigenvalues can make a large problem duration necessary. Therefore it is the spread in Eigenvalues that often causes difficulty for explicit integration methods.

A.3 A Practical Illustration

The next network to be discussed delves a bit more deeply into the practical situation that gives implicit integration an advantage over explicit integration. Consider the circuit in Figure A.1 in which a two-stage transistor circuit drives a large capacitive load. Let the equivalent circuit component data be such that the emitter and collector capacitances are

$$C_E = C_{te} + \Theta_n T_e J_e \quad (\text{A.6})$$

$$C_C = C_{tc} + \Theta_i T_s J_c \quad (\text{A.7})$$

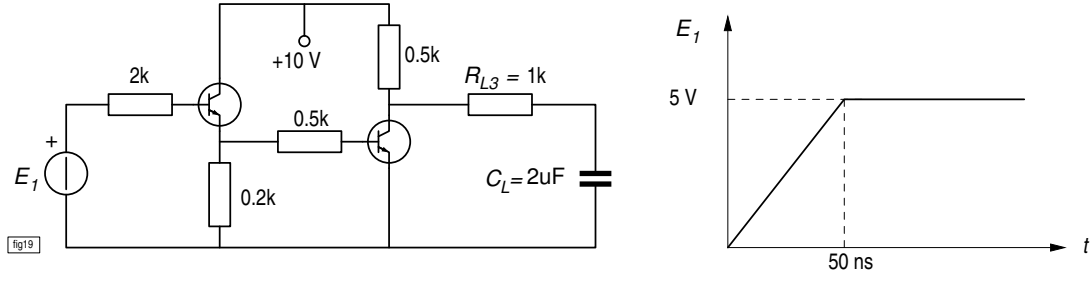


Figure A.1: A Practical Illustration

where

$$\begin{aligned}
 C_{te} &= \text{emitter transition capacitance} = 3 \text{ pF} \\
 C_{tc} &= \text{collector transition capacitance}^1 = 5 \text{ pf} \\
 \Theta_n &= 30 \text{ V}^{-1} \\
 \Theta_i &= 35 \text{ V}^{-1} \\
 T_e &= 0.2 \text{ ns} \rightarrow f_\alpha = 800 \text{ mc} \\
 T_s &= 5 \text{ ns} \\
 J_e &= \text{forward emitter junction current} \\
 J_c &= \text{forward collector junction current}
 \end{aligned}$$

If the input signal contains high frequency components, or if the performance of the individual transistor stages are of interest, the user cannot "cheat" on the equivalent capacitor values. The available component data must be used in Equation (A.6) and (A.7) even if these lead to very small values of capacitance. On the other hand the values used for R_{L3} and C_L lead to an output time constant of 2 ms which in turn requires a problem duration of about 10 ms if the entire transient is to be observed. The familiar eigenvalue spread has been created. When this problem was solved with explicit integration (XPO), 10,000 passes were required to get to a problem time of 58,000 ns — far short of the required duration. Solution with the new implicit method covered the entire transient in 520 passes. With this data one can claim an improvement factor of 3260. This is based on a factor of 19.2 fewer passes to achieve a problem duration of another factor of 170 greater. Entire papers have been written that were based on statistics like this.

The same network can be used to illustrate that there is still a future for explicit integration. If components R_{L3} and C_L are removed from the network, so is the large time constant and any need to run the problem for anything like 10 ms. In fact, 100 ns is more than sufficient to cover the entire transient period. What has been done in effect is to remove the large spread in eigenvalues. This change in computational circumstances also changes the relative efficiencies of the integration methods. Explicit integration now completes the problem in 297 passes while implicit requires 409 passes. Without an eigenvalue spread, the implicit method has lost its large speed advantage and has become about 40 percent slower.

The complete input listing of the circuit of Figure A.1 including the model description is given on page 60. It is interesting to note that the Rerun mode may be used to easily perform comparisons between implicit integration and explicit XPO.

A.4 Eigenvalue spread

A series of runs and reruns were made in order to gather comparative data on the performance of XPO and the implicit method as a function of selected eigenvalue spreads. This information is summarized in Table A.2. With a small spread of ten, it is seen that implicit takes more solution

Table A.2: Explicit-Implicit Comparison at Selected Eigenvalue Spreads

Eigenvalue Spread (approximate)	Explicit (XPO) Passes	Implicit Passes	Improvement Factor
10	113	146	0.77
100	237	171	1.40
1000	703	180	3.90
10000	1807	190	9.50
100000	6664	184	36.00
10^6	16578	185	90.00
10^7	20000*	193	238.00
10^8	20000*	189	996.00

* The last two runs were not carried through the complete transient solution, but were terminated by the program pass limit. The improvement factors for these runs include an extrapolation term that is based on the portion of the run that was not completed.

passes than does explicit — hence an improvement factor of less than unity. As the spreads of eigenvalues are progressively increased, implicit integration becomes comparatively more attractive, and the trend would become more marked if the spread sequence had been continued on.

The forcing function used in all runs was a 1 V step function in order to avoid complicating effects from that direction. If more complex forcing functions were used, it is likely that the improvement factors that are listed in Table A.2 would be modified and in some cases quite significantly. Since this is true, the listed factors should be considered to be valid only as an approximate guide that illustrates the potential of implicit integration under favorable circumstances.

A.5 Jacobian Construction

The need for a Jacobian, or matrix of partial derivatives, arises in the derivation of the corrector iteration of the implicit integration method that has been implemented. This need is common to all implicit methods, both single step and multistep. An m dimensional generalized form of the Jacobian appears in Figure A.2. No different philosophies exist concerning the most efficient method

$$\begin{bmatrix} \frac{\partial F_1}{\partial Y_1}(Y_1, \dots, Y_m, t) & \dots & \frac{\partial F_1}{\partial Y_m}(Y_1, \dots, Y_m, t) \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial Y_1}(Y_1, \dots, Y_m, t) & \dots & \frac{\partial F_m}{\partial Y_m}(Y_1, \dots, Y_m, t) \end{bmatrix}$$

Figure A.2: The General Jacobian

Listing A.1: Circuit Description of figure A.1

```

1  MODEL DESCRIPTION
2  MODEL ZZ (TEMP) (B-C-E)
3  ELEMENTS
4      RB, B-X = 0.3
5      RC, C-Y = 0.015
6      CE, X-E = X1 (3.+6.*JE)
7      CC, X-Y = X2 (5.+175.*JC)
8      JE, X-E = DIODEQ (1.D-7,30.)
9      JC, X-Y = DIODEQ (1.D-7.35.)
10     JX, Y-X= 0.98*JE
11     JY, E-X= 0.1*JC
12  OUTPUTS
13     VCE,VCC,PLOT
14  CIRCUIT DESCRIPTION
15  TWO STAGE CIRCUIT WITH LOAD
16  ELEMENTS
17     E1, 1-2 = TABLE1
18     RB1,2-3 = 2
19     T1-,34-5 = MODEL ZZ
20     E2, 1-4 = 10
21     RL2,4-7 = 0.5
22     RE, 1-5 = 0.2
23     RB2,5-6 = 0.5
24     T2,6-7-1 = MODEL ZZ
25     RL3,7-9 = 1
26     CL, 9-1 = 2E6
27  OUTPUTS
28     VCL,XSTPSZ,PLOT
29  RUN CONTROLS
30     RUN INITIAL CONDITIONS
31     INTEGRATION ROUTINE = IMPLICIT
32     STOP TIME = 1E7
33  FUNCTIONS
34     TABLE1
35         0,0, 50,5, 100,5
36  RERUN DESCRIPTION
37  RUN CONTROLS
38     INTEGRATION ROUTINE = XPO
39     MAXIMIM INTEGRATION PASSES = 10000
40  END

```

of forming this matrix. The first is based on a symbolic approach in which each element of the Jacobian is set up in terms of the literal R, C, L and G components that constitute a given problem. Its advantage lies in the fact that the matrix can be constructed just once and updated as often as required depending on the progress of the solution. Furthermore, the updating process is just a simple matter of data insertion without any need to recompute the system derivatives. When large problems are encountered, sparse matrix techniques can be applied without difficulty. The disadvantage is that some combinations of topologies, dependencies, and element values are encountered such that the Jacobian that is constructed this way is significantly in error. In these cases the average solution step size that can be taken is sometimes significantly reduced and inefficient operation can result.

The second method is purely numerical in nature and can be referred to as "numerical differencing". It produces an approximation to the desired partial derivatives as indicated in Equation (A.8). These approximations are made by making

$$\frac{\partial F_i}{\partial Y_j} \approx \frac{\dot{Y}_i'(Y_1, \dots, Y_m, t) - \dot{Y}_i(Y_1, \dots, Y_m, t)}{Y_j' - Y_j} \quad (\text{A.8})$$

additional derivative computations each time the Jacobian is to be updated. The advantage of this approach is that it is universal in that theoretically it should produce a good approximation to the true Jacobian for any transient problem. Its principal disadvantage is that it requires additional solution passes. The number of additional solution passes will be equal to mX where X is the number of Jacobian evaluations required during the course of a given problem. A second disadvantage is that this method does not readily lend itself to the use of the L-U decomposition sparse matrix technique that is used in the program.

The program will automatically choose between the two methods of Jacobian construction. The procedure that is followed is to categorize all the transient problems that are to be solved with implicit integration into three groups according to the number of state variables contained, m .

Group 1 : $m \leq 10$

Group 2 : $10 < m \leq 50$

Group 3 : $m > 50$.

The program default is set to use the numerical approach for Group 1 problems. The reason for this is that both of the disadvantages of this approach (mX extra derivative evaluations and additional core requirements) are greatly minimized for small problems. Exactly the converse is true for Group 3 problems, so the default choice then becomes the symbolic approach. When the medium size problems represented by the Group 2 classification are encountered, a series of checks will be made. These checks will examine the problem for topological conditions that could lead to error as well as the presence of user supplied differential equations. If any of these checks are positive the numerical approach will be taken. If not, symbolic construction of the Jacobian will be used.

A method has also been provided to allow the user to specify the type of construction to be used which will bypass the default mechanism. If either the numerical or symbolic approach is desired the respective entry under RUN CONTROLS is

```
USE DIFFERENCED JACOBIAN
USE SYMBOLIC JACOBIAN
```

A.6 Preset Step Size Controls

The explicit integration routines in SCEPTRE have preset quantities that control the maximum, minimum, and starting step sizes. These are identical for all of the explicit methods and are as follows:

```
MINIMUM STEP SIZE = 1 × 10-5 (STOP TIME)
MAXIMUM STEP SIZE = 2 × 10-2 (STOP TIME)
STARTING STEP SIZE = 1 × 10-3 (STOP TIME)
```

The preset quantities have been found to be inappropriate for implicit integration and have been replaced with the following:

```
MINIMUM STEP SIZE = 1 × 10-14 (STOP TIME)
MAXIMUM STEP SIZE = 2 × 10-2 (STOP TIME)
STARTING STEP SIZE = 1 × 10-8 (STOP TIME)
```

Any of these may be replaced with a different constant at the discretion of the user with a simple entry under RUN CONTROLS. This format is

```
MINIMUM STEP SIZE = number
MAXIMUM STEP SIZE = number
STARTING STEP SIZE = number
```

The error control differs most markedly from that used for the explicit methods in that the estimated error is not computed and checked for each individual differential equation; but is computed and checked for all of the equations together just once each step. If a given problem consists of m state variables, the following relation is computed at the end of each integration step:

$$D = \left[\frac{E_i}{|Y_i|} \right]^2 + \dots + \left[\frac{E_m}{|Y_m|} \right]^2$$

where the E_i terms designates local error estimates for each differential equation and the Y_i are the state variables with $0 < i \leq m$. The step is accepted if

$$D \leq g \cdot \text{MINIMUM ABSOLUTE ERROR}$$

where g is a constant that depends on the current order of the integration method and MINIMUM ABSOLUTE ERROR has a preset value of 0.001. The nomenclature notwithstanding, this is a relative error control since the magnitude of the state variables enter the computation.

Some testing has been performed to determine the effect of other values for MINIMUM ABSOLUTE ERROR. No significant improvement was found in the inevitable speed-accuracy tradeoff. What was found however, was that larger values which lead to significant speed improvements did so only at the cost of unacceptable degradation in accuracy. The user is free to experiment along these lines by entering under RUN CONTROLS MINIMUM ABSOLUTE ERROR = *number*. The remaining error criteria (MAXIMUM ABSOLUTE ERROR, MAXIMUM RELATIVE ERROR, MINIMUM RELATIVE ERROR) will have no effect on the implicit integration method.

B B Matrix Derivation

The derivation of the general B matrix that expresses link voltages in terms of tree branch voltages and tree branch currents in terms of link currents is as follows:

Two fundamental incidence matrices that arise from network topology theory will be called the Q and T matrices here. The fundamental cut set matrix, $Q = [q_{ij}]$ is a matrix containing $(n - 1)$ rows and b columns for a network containing n nodes and b elements, where:

$q_{ij} = +1$ if the i th fundamental cut set direction coincides with the reference direction of the j th element

$q_{ij} = -1$ if the i th fundamental cut set direction is in opposition to the reference direction of the j th element

$q_{ij} = 0$ if the i th fundamental cut set does not include the j th element.

If the elements are properly ordered, it is always true that

$$Q = [-B^T U]$$

where the columns of the unit matrix U correspond to the tree branch elements.

The fundamental circuit matrix, $T = [t_{ij}]$ is a matrix containing m rows and b columns for a network containing m independent loops and b elements, where

$t_{ij} = +1$ if the i th independent loop direction coincides with the reference direction of the j th element

$t_{ij} = -1$ if the i th independent loop direction opposes the reference direction of the j th element

$t_{ij} = 0$ if the i th independent loop does not include the j th element.

If the elements are properly ordered, it is always true that

$$T = [U B]$$

where the columns of the unit matrix U correspond to the network links. Since it is always true that

$$Q I_b = 0, \quad T V_b = 0$$

direct substitution yields

$$[-B^T U] \begin{bmatrix} I_L \\ I_{TB} \end{bmatrix} = 0 \quad \text{and} \quad [U B] \begin{bmatrix} V_L \\ V_{TB} \end{bmatrix} = 0$$

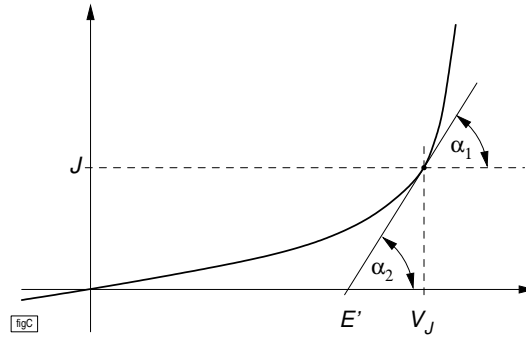
Expansion of these relations gets

$$I_{TB} = B^T I_L \quad \text{and} \quad V_L = -B V_{TB}$$

so that the tree branch currents may be expressed in terms of the link currents and the link voltages may be expressed in terms of the tree branch voltages through the B matrix.

C Diode Representation in the Initial-Conditions Program

The current in a diode or transistor junction at a point on its voltage vs. current (V/I) characteristic may be separated into two components as $J = G_J V_J + Q$. Consider the typical diode curve below.



Angle $\alpha_1 = \alpha_2$ is enclosed by the slope of the diode characteristic at any point and the horizontal at that point. E' is an offset voltage that marks the intersection of a continuation of the slope G_J and the line $J = 0$.

From the figure:

$$\tan \alpha_1 = \tan \alpha_2 = G_J = \frac{J}{V_J - E'}$$

$$\text{or } J = G_J V_J - G_J E'$$

$$\text{or } J = G_J V_J + Q \text{ if } Q \text{ is defined as } -G_J E'$$

D Basic Newton-Raphson Method

Given a single algebraic or transcendental equation of the form $F(x) = 0$, that is single valued and differentiable in the domain of interest, A Newton-Raphson procedure can be constructed using

$$F(x) + \frac{\partial F(x)}{\partial x} \Delta x = 0 .$$

An initial $x = x_0$ may be assumed and $F(x_0), \frac{\partial F(x_0)}{\partial x}$ determined.

Then $\Delta x = \frac{-F(x_0)}{\frac{\partial F(x_0)}{\partial x}}$ and $x_1 = x_0 + \Delta x$.

The procedure is repetitive until

$$|F(x_{n+1}) - F(x_n)| < z$$

where z is some specific convergence criteria. When this last relation is satisfied, the process is said to have converged. Extension to systems of equations add no complications.

E Equivalence of Equations (2.79) and (2.80)

To show the equivalence of equations (2.79) and (2.80), it will be sufficient to show that

$$\begin{bmatrix} V_9 \\ V_2 \\ V_5 \end{bmatrix} - [Z]^{-1} \begin{bmatrix} F_1 & (V_9, V_2, V_5) \\ F_2 & (V_9, V_2, V_5) \\ F_3 & (V_9, V_2, V_5) \end{bmatrix} = [Z]^{-1} \begin{bmatrix} -B_{97}E_7 \\ -B_{27}E_7 \\ [B_{95}^T + B_{05}^T\alpha]Q_9 + B_{85}^TJ_8 \end{bmatrix}$$

or

$$\begin{bmatrix} V_9 \\ V_2 \\ V_5 \end{bmatrix} = [Z]^{-1} \left\{ \begin{bmatrix} -B_{97}E_7 \\ -B_{27}E_7 \\ [B_{95}^T + B_{05}^T\alpha]Q_9 + B_{85}^TJ_8 \end{bmatrix} + \begin{bmatrix} F_1 & (V_9, V_2, V_5) \\ F_2 & (V_9, V_2, V_5) \\ F_3 & (V_9, V_2, V_5) \end{bmatrix} \right\}$$

or

$$[Z] \begin{bmatrix} V_9 \\ V_2 \\ V_5 \end{bmatrix} = \begin{bmatrix} -B_{97}E_7 \\ -B_{27}E_7 \\ [B_{95}^T + B_{05}^T\alpha]Q_9 + B_{85}^TJ_8 \end{bmatrix} + \begin{bmatrix} F_1 & (V_9, V_2, V_5) \\ F_2 & (V_9, V_2, V_5) \\ F_3 & (V_9, V_2, V_5) \end{bmatrix}.$$

The left side of the equation expands into

$$\begin{bmatrix} V_9 + B_{95}V_5 \\ V_2 + B_{25}V_5 \\ -[B_{95}^T + B_{05}^T\alpha]G_{99}V_9 - B_{25}^TG_{22}V_2 + G_{55}V_5 \end{bmatrix}$$

and the right side, upon substitution of equations (2.71) through (2.74) becomes

$$\begin{bmatrix} V_9 + B_{95}V_5 \\ V_2 + B_{25}V_5 \\ -[B_{95}^T + B_{05}^T\alpha]G_{99}V_9 - B_{25}^TG_{22}V_2 + G_{55}V_5 \end{bmatrix}$$

which shows the equivalence.

F Adjoint Network for Sensitivity

A number of papers [19, 20] have been written on determining the unnormalized sensitivity (i.e., partial derivatives) of network functions with respect to network elements by an adjoint network. To find the sensitivity of any network function by this method, two analyses are required, one of the given network and the other of the related adjoint network. This appendix describes the generation of the adjoint network for DC calculations in SCEPTRE. A detailed derivation of the adjoint network for sensitivity is given in References [19, 20]. Due to the sign conventions followed in SCEPTRE, the terms given in tables F.1 and F.2 differ, in sign, from those given by Director and Rohrer [19, 20]. Table F.1 describes the relation between a given network and its related adjoint network. It also describes the allowed dependent variables (network functions) and independent variables (network elements). Table F.2 describes the forcing functions for the adjoint network and the sensitivities. The superscript "a" in the tables indicates the currents and voltages in the adjoint network.

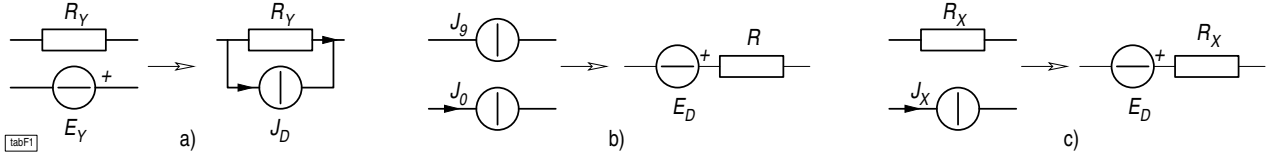


Figure F.1: Relation between Network and Adjoint Network (see explanation in table F.1)

Secondary current sources in the given network introduce voltage dependent voltage sources in the adjoint network (shown in Table F.1). This requires modification of the Jacobian. Derivation of the modified Jacobian follows. Let us classify the voltage sources, arising in the adjoint calculation due to secondary sources (J_0) in the main circuit, as E_D .

The B matrix from the L tree which includes the E_D 's is as shown below.

	Class C_4	Class R_5	Class L_6	Class E_7	Class E_D
Class C_1	B_{14}	B_{15}	B_{16}	B_{17}	B_{1D}
Class R_2	0	B_{25}	B_{26}	B_{27}	B_{2D}
Class L_3	0	0	B_{36}	B_{37}	B_{3D}
Class J_8	B_{84}	B_{85}	B_{86}	B_{87}	B_{8D}
Class J_9	B_{94}	B_{95}	B_{96}	B_{97}	B_{9D}
Class J_0	B_{04}	B_{05}	B_{06}	B_{07}	B_{0D}

The following equations arise from the above B matrix if vectors V_6, V_3, I_4, I_1 and submatrix B_{94} are assumed to be zero. These assumptions are based on the known final values of V_6, V_3, I_4, I_1 for the initial condition problem and on the absence of any current-source and capacitor cut sets.

$$I_5 - B_{25}^T I_2 - B_{85}^T J_8 - B_{95}^T J_9 - B_{05}^T J_0 = 0 \quad (\text{F.1})$$

$$V_2 + B_{25} V_5 + B_{27} E_7 + B_{2D} E_D = 0 \quad (\text{F.2})$$

$$V_9 + B_{95} V_5 + B_{97} E_7 + B_{9D} E_D = 0 \quad (\text{F.3})$$

Table F.1: Relation between Network and Adjoint Network

Network Functions (i.e., Dependent Variables)		Network Elements (i.e., Independent Variables)	
Allowed for Sensitivity			
Voltages		Currents	
Yes	Yes	Yes	Yes
Yes	Not Allowed	Not Allowed	Not Allowed
Not Allowed	Yes	Yes	Not Allowed
Not Allowed	Not Allowed	Not Allowed	Not Allowed
Not Allowed	Yes	Yes	Yes
Not allowed	Yes (IE_Y)	Yes	Not Allowed

Given Network Elements	Adjoint Network Elements	
Resistors R	Resistors R	
Capacitors C	Capacitors C	
Inductors L	Inductors L	
Mutuals M	Mutuals M	
Independent Voltage Sources E_7	Set to Zero, i.e. $E_7 = 0$	
Linearly Dependent Voltage Sources E_Y (see fig. F.1 a)	Set E_Y to zero and add a dependent current source across R_Y (see fig. F.1 a)	
$E_Y = \mu \cdot VR_Y$	$J_D = \mu \cdot IE_Y^a, \quad E_Y^a = 0$	

Independent Current Sources J_8	Set to zero, i.e., $J_8 = 0$	Yes	Not Allowed	Yes
Primary Current Sources J_9	Converted to resistors where $R =$ slope of diode characteristic	Yes	Yes	Not Allowed
Secondary Current Sources J_0 (see fig. F.1 b)	J_9 is converted to resistor as above and a dependent series voltage source is added and J_0 is set to zero (see fig. F.1 b); $R =$ slope of diode characteristic	Yes (VJ_0)	Not Allowed	Yes, α given by defined parameter
$J_0 = \alpha \cdot J_9$	$E_D = \alpha \cdot V_{J_0}^a, \quad J_0^a = 0$			

Linearly Dependent Source J_X (see fig. F.1 c)	Set J_X to zero and a dependent voltage source in series with R_X (see fig. F.1 c)	Yes	Not Allowed	Not Allowed
$J_X = \beta \cdot I_{RX}$	$E_D = \beta \cdot V_{J_X}^a, \quad J_X^a = 0$			

Defined Parameters P	Defined Parameters P	P which are function of above allowable network functions and whose differentials are given by G_P	Yes, where P is used in J_0 specification
------------------------	------------------------	--	---

Table F.2: Adjoint Network Applied Source

Dependent Variable Type:	Resistor Voltage, Capacitor Voltage, Voltage across current source	Resistor current, Inductor current, Current thru voltage source, current thru primary dependent current source
Adjoint Source Applied:	Current source of value -1 in parallell with element whose voltage is dependent variable	voltage source of value -1 in series with element whose current is dependent variable
Computation of $\frac{\partial(\text{dependent var})}{\partial(\text{independent var})}$ by independent variable typ:		
Resistor R		$I_R \cdot I_R^a$
Independent voltage source E_7		$-I_{E7}^a$
Independent current source J_8		V_{J8}^a
Defined parameter relating primary & secondary current source $J_0 = P \cdot J_9$		$J_9 \cdot (-V_{J0}^a)$

To express Equation (F.1), (F.2) and (F.3) in terms of V_5, V_2, V_9, J_8, E_7 the following equations are required:

$$I_5 = G_{55}V_5 \quad (\text{F.4})$$

$$I_2 = G_{22}V_2 \quad (\text{F.5})$$

$$J_9 = G_{99}V_9 + Q_9 \quad (\text{F.6})$$

$$J_0 = \alpha J_9 \quad (\text{F.7})$$

For the definition of $G_{55}, G_{22}, \alpha, G_{99}$ and Q_9 see subsection 2.4.1. Two more equations required for complete derivations are (F.8) and (F.9) below.

$$E_D = \alpha^* V_{J0} \quad (\text{F.8})$$

where α^* is the diagonal matrix whose (i, i) element is the non-zero entry of the i th row of Equation (F.7), i. e. it is the coefficient of the J_9 on which the i th J_0 depends.

The absence of any current-source cut set in the given circuit makes B_{0D} equal to zero. This gives

$$V_{J0} + B_{05}V_5 + B_{07}E_7 = 0. \quad (\text{F.9})$$

From Equation (F.1) through (F.9), we get

$$G_{55}V_5 - B_{25}^T G_{22}V_2 - B_{85}^T J_8 - [B_{95}^T + B_{05}^T \alpha][G_{99}V_9 + Q_9] = 0$$

$$V_2 + [B_{25} - B_{2D}\alpha^* B_{05}]V_5 + [B_{27} - B_{2D}\alpha^* B_{07}]E_7 = 0$$

$$V_9 + [B_{95} - B_{9D}\alpha^* B_{05}]V_5 + [B_{97} - B_{9D}\alpha^* B_{07}]E_7 = 0$$

Following through a derivation similar to that done in subsection 2.4.1 we get Equation (F.10). Equation (F.11) is the original SCEPTRE derivation.

$$\begin{Bmatrix} V_{5(n+1)} \\ V_{2(n+1)} \\ V_{9(n+1)} \end{Bmatrix} = \begin{Bmatrix} G_{55} & -B_{25}^T G_{22} & [B_{95}^T + B_{05}^T \alpha] G_{99} \\ [B_{25} - B_{2D} \alpha^* B_{05}] & I & 0 \\ [B_{95} - B_{9D} \alpha^* B_{05}] & 0 & I \end{Bmatrix}^{-1} \begin{Bmatrix} [B_{95}^T + B_{05}^T \alpha] Q_9 + B_{85}^T J_8 \\ [-B_{27} + B_{2D} \alpha^* B_{07}] E_7 \\ [-B_{97} + B_{9D} \alpha^* B_{07}] E_7 \end{Bmatrix} \quad (\text{F.10})$$

$$\begin{Bmatrix} V_{5(n+1)} \\ V_{2(n+1)} \\ V_{9(n+1)} \end{Bmatrix} = \begin{Bmatrix} G_{55} & -B_{25}^T G_{22} & -[B_{95}^T + B_{05}^T \alpha] G_{99} \\ B_{25} & I & 0 \\ B_{95} & 0 & I \end{Bmatrix}^{-1} \begin{Bmatrix} [B_{95}^T + B_{05}^T \alpha] Q_9 + B_{85}^T J_8 \\ -B_{27} E_7 \\ -B_{97} E_7 \end{Bmatrix} \quad (\text{F.11})$$

Comparison of Equation (F.10) and (F.11), shows that terms get added to the (2,1) and (3,1) terms of the Jacobian and to the forcing function column vector of equation (F.10), while the term originally involving α are deleted.

Therefore when an adjoint run is requested, and secondary sources are present in the given circuit, the iteration given by Equation (F.10) is used instead of the one given by (F.11).

Bibliography

- [1] D. Becker. *Extended SCEPTRE*. Mathematical Formulation. AFWL: TR-73-75, NTIS: ADA 009595.
- [2] D. Becker. *Extended SCEPTRE*. User's Manual. AFWL: TR-73-75, NTIS: ADA 009594.
- [3] J.C. Bowers and S.R. Sedore. *SCEPTRE: A Computer Program for Circuit and System Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1971.
- [4] J. Portasik and C. Glass. *System Analysis Using the SCEPTRE Computer Program*. 98. AFWL, Sept. 1968.
- [5] S. R. Sedore. "SCEPTRE: A Program for Automatic Network Analysis". In: *IBM Journal of Research and Development* 2 (Nov. 1967).
- [6] S. R. Sedore. "More Efficient Use of the F Matrix in Practical Circuit Analysis Program". In: *IEEE Trans. on Computers* 17 (May 1968).
- [7] M. Fowler and R. Warten. "A Numerical Integration Technique for Ordinary Differential Equations with Widely Separated Eigenvalues". In: *IBM Journal of Research and Development* 2 (Sept. 1967).
- [8] R. H. Dickhaut. *Advances in Computer Techniques for Radiation Effects Calculation*. D2-90571. Boeing, 1964.
- [9] W. C. Cordwell. *Radiation Device Modeling for the SCEPTRE Program*. 69-825-2371. Owego, N.Y.: IBM, June 1969.
- [10] M. B. Reed and S. Seshu. *Linear Graphs and Electrical Networks*. Reading, Mass.: Addison-Wesley, 1961.
- [11] F. B. Hildebrand. *Introduction to Numerical Analysis*. New York: McGraw-Hill, 1956.
- [12] R. M. Warten. "Automatic Step-Size Control for Runge-Kutta Integration". In: *IBM Journal of Research and Development* 7.4 (Oct. 1963), pp. 340–341.
- [13] W. Kaplan. *Ordinary Differential Equations*. Reading, Mass.: Addison Wesley, 1958. Chap. 10-2, pp. 400–401.
- [14] W. Kaplan. *Ordinary Differential Equations*. Reading, Mass.: Addison Wesley, 1958. Chap. 10-4, "Heun's Method", pp. 402–403.
- [15] K. G. Ashar, H. N. Ghosh, A. W. Aldridge, and L. J. Patterson. "Transient Analysis and Device Characterization of ACP Circuits". In: *IBM Journal of Research and Development* 7 (1963), p. 218.
- [16] P. E. Chase. "Stability Properties of Predictor-Corrector Methods for Ordinary Differential Equations". In: *Journal A.C.M* 9 (Oct. 1962), pp. 457–468.
- [17] C. W. Gear. "The Automatic Integration of Ordinary Differential Equations". In: *Comm. of A.C.M.* 14 (Mar. 1971).
- [18] C. W. Gear. "The Numerical Integration of Ordinary Differential Equations". In: *Math. Comp.* 21 (1967).

- [19] S. W. Director and R. A. Rohrer. “The Generalized Adjoint Network and Network Sensitivities”. In: *IEEE Transaction on Circuit Theory* 16 (Aug. 1969).
- [20] S. W. Director and R. A. Rohrer. “Automated Network Design – The Frequency-Domain Case”. In: *IEEE Transaction on Circuit Theory* 16 (Aug. 1969).
- [21] Wm. C. Davidon. *Variable Metric Method for Minimization*. Report ANL-5990 (REV). AEC Research and Development, 1959.
- [22] M. J. Box. “A Comparison of Several Current Optimization Methods, and the Use of Transformation in Constrained Problems”. In: *The Computer Journal* 9 (1966), pp. 67–77.
- [23] M. J. Box, D. Davies, and W. G. Svann. *Non-Linear Optimization Techniques*. Monograph No. 5. Imperial Chemical Industries Ltd., 1969.
- [24] *Variable Metric Minimization*. SHARE Routine No. 1117, AN Z013, A3. Suite 750, 25 Broadway New York, NY 10004: SHARE Inc.
- [25] W. Tinney and J. Walker. *Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization*. IEEE, Nov. 1967.
- [26] J. Grad and M. A. Brebner. “Algorithm 343, Eigenvalues and Eigenvectors of a general matrix”. In: *Comm. ACM* 11 (Dec. 1968), pp. 820–826.
- [27] J. V. Leeds and G. I. Ugron. “Simplified Multiple Parameter Sensitivity Calculation and Continuously Equivalent Network”. In: *IEEE Trans. on Circuit Theory* 14 (June 1967).

AFWL: Air Force Weapons Laboratory, Kirtland, AFB, NM 87117, USA

NTIS: National Technical Information Service, Springfield, VA 22151, USA

AFAPL: Air Force Aero Propulsion Laboratory, Wright-Patterson AFB, Ohio 45433, USA

Index

absolute error criterion, [20](#)
backward Euler technique, [56](#)
Davidon algorithm, [32](#)
error criterion, [20](#)
implicit Euler technique, [56](#)
implicit trapezoidal integration, [57](#)
Jacobian, [24](#)
modal matrix, [36](#)
Newton-Raphson Method, [32](#)
relative error criterion, [20](#)
Steepest Descent Method, [32](#)