SY

SYMBOLIC ANALOG CIRCUIT ANALYSIS

By

Sin-Min Chang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1992

ABSTRACT

SYMBOLIC ANALOG CIRCUIT ANALYSIS

By

Sin-Min Chang

Studies have shown that the efficiency of symbolic circuit analysis can be improved by a composition and decomposition strategy. However, most of the existing strategies are not suitbale for analyzing commercial ICs. This limits the application of symbolic approach for circuit analysis. In this dissertation, a new circuit level decomposition strategy is presented. Taking advantage of the characteristics of the nullors, computing efforts can be further reduced by exploring valid and invalid decomposed sub-circuits. Together with a new numerical approximation, the symbolic sensitivity analysis, and the symbolic Hurwitz test capabilities, Sspice, a symbolic circuit analyzer, is capable of analyzing circuits like commercial op-amps and active filters. Examples are given.

This research was supported in part by REF of State of Michigan

LIST O

LIST O

1 MO
1.1
1.2

2 SY
2.1
2.2

2.

2.

3 D
A
3
3
3

3
3

4 I
I
4

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

1 MOTIVATION 1

1.1 Analog Circuit Design Process 1

1.2 Analog Circuit Analysis 3

2 SYMBOLIC CIRCUIT ANALYSIS 11

2.1 Introduction 11

2.2 Circuits with Nullors 13

2.2.1 Nullator-Norator Nodal Analysis 13

2.2.2 Equivalence relations 19

2.2.3 Special Case for Network Determinants 19

2.3 Graph Theory Approach 23

2.3.1 Coates Graph and Tree-Enumeration Method 24

2.3.2 Signal-Flow Graph and Mason's Rule 26

2.4 The Existing Symbolic Circuit Analyzers 30

3 DECOMPOSITION STRATEGIES FOR SYMBOLIC CIRCUIT ANALYSIS 33

3.1 A Review of Decomposition Methods 33

3.2 Obtaining Network Determinants by Decomposition 35

3.3 The Algorithm 39

3.3.1 Analysis 46

3.4 Exploiting Special Decomposition Opportunities 49

3.5 A Circuit Level Decomposition Application 55

4 NUMERICAL APPROXIMATION STRATEGIES FOR SYMBOLIC CIRCUIT ANALYSIS 66

4.1 Numerical Approximation After Computation 67

4.
4.
4.
4.2 N
4.3 N

5 SYMB
5.1 T
5.2 A

6 SYMB
6.1 H
6.2 I

7 IMPL
7.1 M
7.2 O
7.3 I
7.4 S
7.5 I
7.6 S

8 CON

BIBLIO

4.1.1 Numerical Approximation in ISAAC 70
4.1.2 Numerical Approximation in SCOPE 71
4.1.3 Numerical Approximation in Sspice 72
4.2 Numerical Approximation During Computation 72
4.3 Numerical Substitution Before Computation 75

5 SYMBOLIC SENSITIVITY ANALYSIS 77
5.1 The Implementation of Symbolic Sensitivity Analysis 77
5.2 Applications of Sensitivity Analysis 79

6 SYMBOLIC STABILITY ANALYSIS 93
6.1 Hurwitz Test Fundamentals 93
6.2 Implementation of Hurwitz Test 96

7 IMPLEMENTATION OF SSPICE VERSION 2.0 101
7.1 Matrix Reduction Method 101
7.2 Obtaining Matrix Determinant 104
7.3 Implementation of Sspice 107
7.4 Second Order Filter Function Identification 113
7.5 In-Band Error Approximation 115
7.6 Special Functions 118

8 CONCLUSIONS 122

BIBLIOGRAPHY 125

LIST OF TABLES

1.1 Characteristics of Analog and Digital Circuit Designs. 2

3.1 Symbolic Network Determinants of Example 4. 54
3.2 *o* and *u* indexes. 57

4.1 With and Without Approximation for P1 and P2 75

LIST OF FIGURES

1.1 Flow chart for circuit design process. 2
1.2 Series RLC circuit. 5
1.3 Bandstop filter function at V2 of series RLC circuit. 5
1.4 (a) CMOS op-amp. (b) Open Loop gain circuit. 7
1.5 Transfer function of the CMOS op-amp. 8

2.1 (a)Nullator; (b)Norator. 12
2.2 Nullator-norator equivalent circuit of a VCVS 12
2.3 A circuit with nullator and norator. 13
2.4 Numerator of V_4 in Example 1. 16
2.5 The equivalent network determinant of $\mathbf{Y}_{1,1}$. 16
2.6 Parallel and Open/Short equivalence. 20
2.7 (a)Nullator trees; (b)Norator trees. 20
2.8 A network has two parts connecting at a single node. 22
2.9 A corresponding Coates graph. 26
2.10 A typical Mason graph 28

3.1 Example 2 36
3.2 The decomposition and composition of two sub-circuits connected at three nodes; (a) The whole circuit; (b)-(g) six cases. 40
3.3 Example with Na and Nb 44
3.4 Maximum beneficial cut nodes analysis. 48
3.5 Example 3. 51
3.6 Example 4. 55
3.7 An Example Circuit with two parts. 56
3.8 The Equivalent Circuit 56
3.9 Cases expanded with respect to `G1`. (a) `ND1a`, Network Determinant without `G1`. (b) `ND1`, Further Simplification of (a). (c) `ND2`, Network Determinant with `G1`. 59

3.10 Cases expanded with respect to G2 of Figure 16(b). (a) ND11a, Network Determinant without G2. (b) ND11, Further Simplification of (a). (c) ND12, Network Determinant with G2. 60
3.11 Cases expanded with respect to G2 of Figure 16(c). (a) ND21, Network Determinant without G2. (b) ND22, Network Determinant with G2. . . 61
3.12 Network Determinant Equation of obtaining transfer function 62
3.13 α, β network representation of the transfer function. 63
3.14 Schematic diagram of the transfer function **K**. 64
3.15 Schematic diagram of the transfer function α. 65
3.16 Schematic diagram of the transfer function β. 65

4.1 A Example for Approximation 74

5.1 Tow-Thomas Active Filter 80
5.2 Bandpass filter function of V2. 80
5.3 Sensitivity of V2 with respect to R3. 81
5.4 Wo of Tow-Thomas Filter. 85
5.5 Sensitivity of Wo of Tow-Thomas Filter. 85
5.6 Qo of Tow-Thomas Filter. 86
5.7 Sensitivity of Qo of Tow-Thomas Filter. 86
5.8 State Variable Active Filter 87
5.9 Wo of the State Variable Active Filter. 88
5.10 Qo of the State Variable Active Filter. 88
5.11 Sensitivity analysis without approximation 90
5.12 Sensitivity analysis with approximation 90
5.13 Bandpass filter function of V6 in Svaf 91

7.1 Series resistor circuit. 102
7.2 Relations between minors. 105
7.3 Flow Chart of Sspice. 108
7.4 The SPICE file of 741 chip without compensation. 110
7.5 Frequency response of uA741 op-amp for gain and phase. 111
7.6 Sspice output of open loop transfer function of 741 without compensation. 112
7.7 (a) Low Pass. (b) High Pass. (c) Band Pass. (d) Band Stop. (e) All Pass filter functions. 114
7.8 Pspice simulation of State Variable Active Filter using ideal and non-ideal op-amps. 119

CH

MO

Due t

becom

roll in

still i

of cir

in-de

circu

circu

C

level

simu

tools

1.1

A ci

CHAPTER 1

MOTIVATION

Due to the advances of integrated circuit technologies, computer-aided design has become essential for system development. For years, CAD has played an important roll in designing digital systems. However, the CAD technology for analog circuits is still in its infancy. The major difficulty is the lack of understanding of the diversities of circuits. Therefore, it is very difficult to predict the behavior of a system without in-depth analysis. Directly imposing the design process of digital circuits on analog circuits is impractical. Table 1.1 [1] shows the differences between analog and digital circuit designs.

Currently, the design and synthesis of analog circuits depends heavily on circuit level simulation. The ability of each circuit designer to analyze the results from simulation determines the performance and characteristics of his/her design. CAD tools for circuit analysis become necessary for assisting analog circuit design.

1.1 Analog Circuit Design Process

A circuit design process for analog circuits is illustrated in Figure 1.1.

Analog Design	Digital Design
Signals have a range of values for amplitude and time	Signals have only two states
Irregular blocks	Regular blocks
Customized (Semicustomized approach is under development)	Standardized
Components have a range of values	Components with fixed values
Requires precise modeling	Modeling can be simplified
Difficult to use with CAD	Amenable to CAD methodology
Designed at the circuit level	Designed at the system level
Longer design times	Short design times
Two or three tries necessary for success	Successful circuits the first time
Difficult to test	Amenable to design-for-test

Table 1.1. Characteristics of Analog and Digital Circuit Designs.

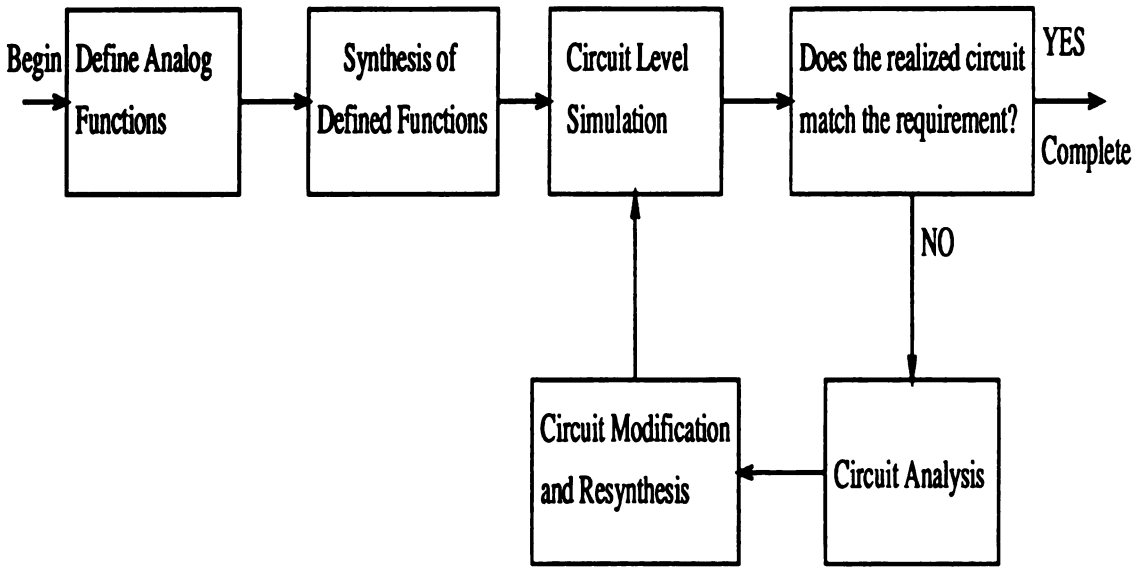


Figure 1.1. Flow chart for circuit design process.

1. Define

the s

2. Synth

whic

3. Circu

ized

cor

3. Circ

th

fu

4. Cir

a

At

simula

at a r

time,

neede

is sti

differ

still

1. **Define Analog Functions :** The designer has to define the functions needed for the specific application.

2. **Synthesis of Defined Functions :** Circuit designer has to choose a realization which fits the requirement under a reasonable approximation.

3. **Circuit Level Simulation :** Circuit simulators are used to verify that the realized circuit does reach the defined function. This simulation should accurately compare with the characteristics of a real product.

3. **Circuit Analysis :** If the realized circuit does not meet the specification, analyze this circuit to find the main contributor to the deviation from the designed function.

4. **Circuit Modification and Resynthesis :** Modify and resynthesize the circuit according to the information obtained from circuit analysis. Then repeat the procedure in item 3.

At present, much effort is being placed in developing circuit simulators. Circuit simulators are now accessible to most designers so that their designs can be verified at a reasonable cost. However, it is difficult to synthesize a circuit correctly the first time, even for an experienced circuit designer. Further modifications are inevitably needed in most cases. How to acquire the information needed for the modifications is still a problem. Many times, it is done by multiple runs of the simulator for a different set of element values. Therefore, experience, intuition, and good luck are still the major factors in this process. They are all ad hoc approaches.

1.2 Analog Circuit Analysis

The most common circuit analysis methods incorporated into numerical circuit simulators are DC, AC, and Transient analysis.

Basically, DC analysis determines the dc operating point of the circuit with inductors shorted and capacitors opened. Usually, DC analysis is performed prior to a transient analysis to determine the transient initial conditions, and prior to an ac small-signal analysis to determine the linearized small-signal models for nonlinear devices.

The results of AC analysis report the ac output variables as a function of frequency. AC analysis computes the dc operating point of the circuit and determines linearized small-signal models for all of the nonlinear devices in the circuit. The resultant linear circuit is then analyzed over the specified range of frequencies. Usually, the input of the linearized circuit is set to one, so that the output of the circuit becomes its transfer function.

The Transient analysis computes the output variables as a function of time. In other words, it performs the simulation of the specified circuit with respect to the inputs in time domain.

Upon understanding these circuit analysis techniques, the next question would be whether these numeric approaches are sufficient to provide all the informations needed for modifying a circuit. This can be illustrated through the following examples.

Example 1 *Figure 1.2 shows the schematic diagram of a series RLC circuit, where the transfer function at node 2 is a notch filter. The stopband frequency is about 50K Hz; and the Qo is about 0.673. Is there any way that a circuit designer can modify this circuit so that the stopband frequency is moved to 100K Hz without affecting the Qo?*

Answer: Usually, a circuit designer understands that the inductor and the capacitor control the stopband frequency. The designer may change the values of the capacitor or the inductor to move the W_z at node 2. Then, he/she may use a

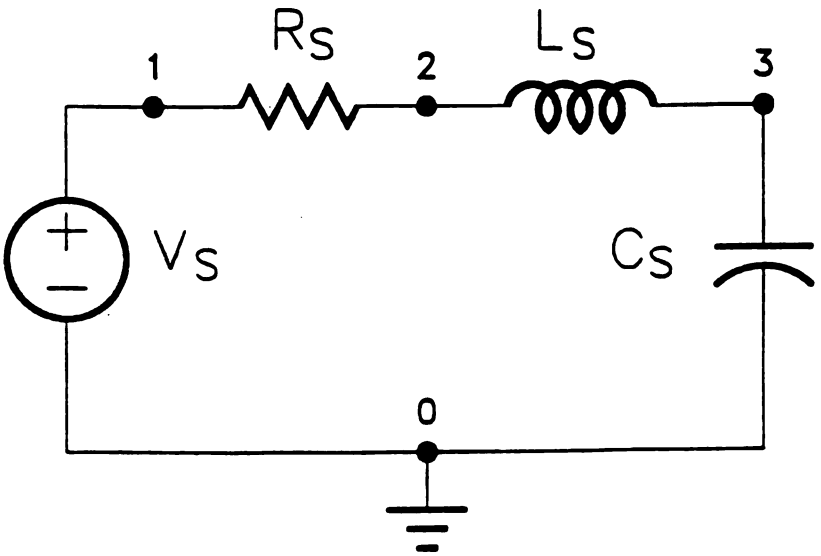


Figure 1.2. Series RLC circuit.

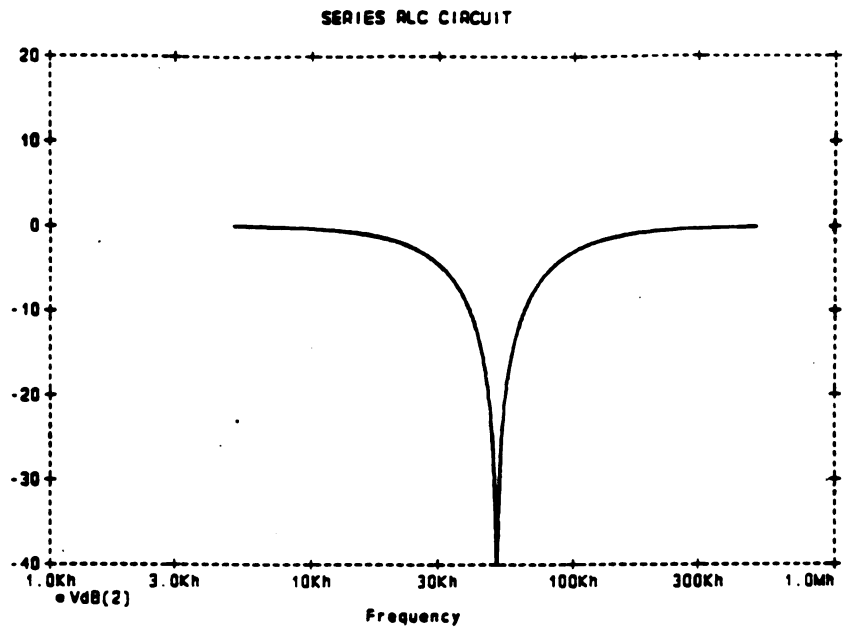


Figure 1.3. Bandstop filter function at V2 of series RLC circuit.

numeri

the Qo

the mo

satisfie

Alt

node 2

The fo

Ther

the r

Exa

the

in F

Is t

GB

cap

amp

spe

numerical circuit simulator to verify the modification. The modification may affect the Qo unintentionally. The values are repeatedly adjusted by trial and error and the modifications repeatedly verified using the simulator until the specifications are satisfied.

Alternatively, an experienced circuit designer may derive the transfer function at node 2, which is

$$\frac{V_2}{V_1} = \frac{(\mathtt{LS} \times \mathtt{CS} \times \mathtt{GS})s^2 + \mathtt{GS}}{(\mathtt{LS} \times \mathtt{CS} \times \mathtt{GS})s^2 + (\mathtt{CS})s + \mathtt{GS}}.$$

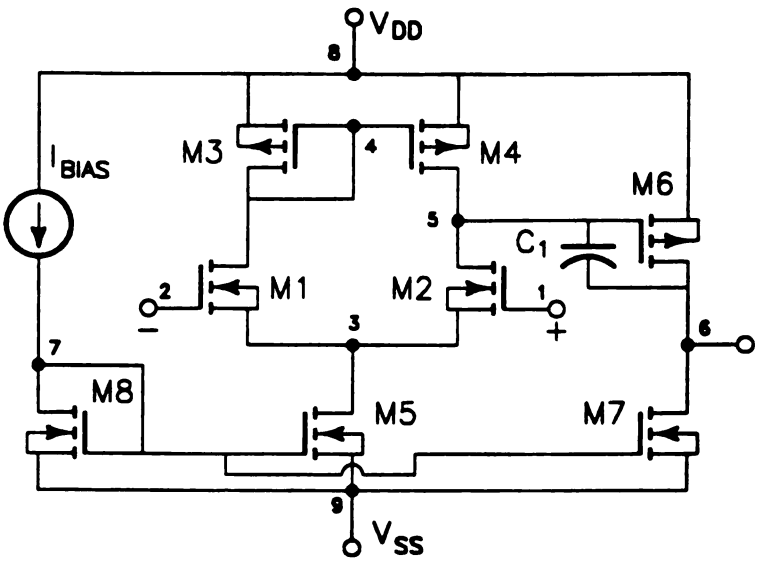
The formulas of Q_o and W_z are, thus, found to be

$$\begin{aligned} Q_o &= \mathtt{GS} \times \sqrt{\frac{\mathtt{LS}}{\mathtt{CS}}}, \textit{and} \\ W_z &= \frac{1}{\sqrt{\mathtt{LS} \times \mathtt{CS}}}. \end{aligned}$$

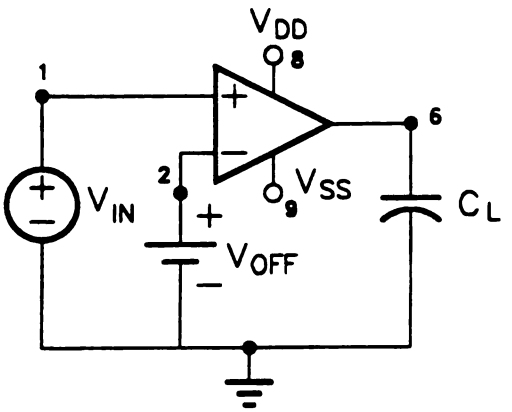
Therefore, this problem can be solved by increasing the value of LS × CS while keeping the ratio between LS and CS. □

Example 2 *Figure 1.4 shows a CMOS op-amp and the schematic diagram to obtain the transfer function. Figure 1.5 shows the transfer function of this op-amp. The* CL *in Figure 1.4(b) is the parasitic capacitor from the circuitry connected to this op-amp. Is there any way that the stability of op-amp can be improved without affecting the GBW?*

Answer: Figure 1.5 shows the transfer function of the CMOS op-amp with a capacitive load. It is known that a capacitive load can affect the stability of an op-amp circuit dramatically. A circuit designer who designs the op-amp which fits the specific application may not be able to change the circuits that this op-amp connects



(a)



(b)

Figure 1.4. (a) CMOS op-amp. (b) Open Loop gain circuit.

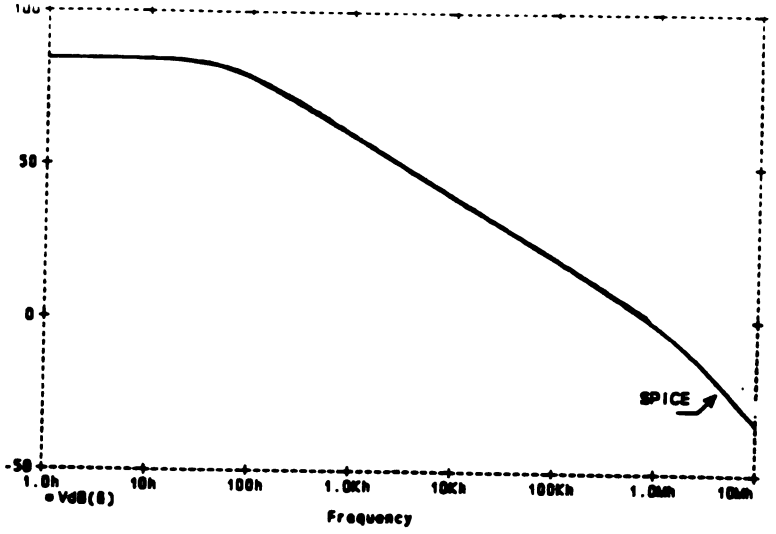


Figure 1.5. Transfer function of the CMOS op-amp.

connects to. What he/she can do is to understand the effect of the capacitive load and modify the internal components of the op-amp so that it can be stabilized for this specific configuration. However, the numerical analysis like Figure 1.5 gives very little about how to improve this op-amp.

If a formula like

$$\begin{aligned}\frac{V_6}{V_1} &= -\frac{sC_1 \times GM_1 - GM_6 \times GM_1}{s^2 C_L \times C_1 + sC_1 \times GM_6 + (GDS_7 + GDS_6)(GDS_3 + GDS_1)} \\ &= \frac{K(s+z_1)}{(s+p_1)(s+p_2)}\end{aligned}$$

is available, assuming $GM_1 = GM_2$ and $GM_3 = GM_4$ for symmetry, then the zero and poles can be obtained.

$$\begin{aligned}z_1 &= -\frac{GM_6}{C_1}, \\ p_2 &= \frac{GM_6}{C_L}, \\ p_1 &= \frac{(GDS_7 + GDS_6)(GDS_3 + GDS_1)}{C_1 \times GM_6},\end{aligned}$$

$$\text{dc gain} = \frac{GM_6 \times GM_1}{(GDS_7 + GDS_6)(GDS_3 + GDS_1)}.$$

The location of p_2 represents the stability of the op-amp, which is determined by transistor 6 and the load capacitor. The circuit designers can change the W/L ratio of the MOS transistor to modify the value of GM_6. Also,

$$GBW = \text{dc gain} \times p_1 = \frac{GM_1}{C_1},$$

which is independent of GM_6. □

The above example shows that a numerical circuit simulator may not fulfill the needs for circuit analysis. In order to improve the quality of analog circuit design, a symbolic circuit analysis methodology providing the analytic solution to the circuits is developed. Without a symbolic circuit analyzer, it is very difficult for circuit designers to have an in-depth understanding of a complicated circuit. However, there are difficulties involved in symbolic circuit analysis. These problems are addressed in Chapter 2.

The most common approaches to improve efficiency are by way of circuit decomposition. Many approaches have been proposed. However, the existing methods are not suitable for practical electronic subsystems like the op-amp and the power supply regulator. A new circuit level decomposition method and its variations are, therefore, developed and described in Chapter 3. Chapter 4 introduces a new numerical approximation strategy. This strategy improves the memory consumption of a symbolic circuit analyzer.

Besides having an efficient mathematical method for symbolic computation, it is also essential to have built-in functions to help the circuit designers perform analysis. Therefore, symbolic sensitivity analysis and symbolic stability analysis are introduced

in Cha

in Cha

process

tion ar

in secc

Fir

in Chapter 5 and Chapter 6. The implementation of *Sspice version 2.0* is introduced in Chapter 7. In addition, second order filters are the most popular circuits in signal processing. The implementation and application of second order function identification are discussed in Chapter 7. Also, the error analysis due to a non-ideal op-amp in second order filter implementation is discussed in the same chapter.

Finally, the conclusions are given in Chapter 8.

CHAPTER 2

SYMBOLIC CIRCUIT ANALYSIS

2.1 Introduction

The importance of a symbolic circuit analyzer has been recognized by circuit designers, since the numerical circuit simulator, alone, cannot give insight into the behavior of an analog circuit [2] [3]. This has lead to the development [4] [5] of various symbolic analog circuit analyzers. Usually, symbolic circuit analysis involves finding network equations in the form of

$$H(s,X) = \frac{N(s,X)}{D(s,X)},$$

where N(s,X) and D(s,X) are polynomials in s and the symbolic network variables X. The method used [4] [5] could be the tree enumeration method, numerical interpolation method, parameter extraction method, signal flow graph method, algebra method or iterative method. The common difficulties inherent in symbolic circuit analyzers are their level of inefficiency for obtaining the circuit functions as compared to its numerical counterpart, their memory space consumption, and the interpretability of their results. These problems are addressed in this chapter.

Nulla

[6]. Fig

current

possible

On the

allows

equatic

Nullators and norators are interesting circuit primitives introduced in the 60's [6]. Figure 2.1 shows their symbols. A nullator is an element which does not allow current flow through it and the voltage across its terminals is zero under all the possible situations. The element is thus described by two equations :

$$V = 0; I = 0. \tag{2.1}$$

On the other hand, the norator has an arbitrary voltage across it and , simultaneously, allows an arbitrary current to flow through it. This element has no constitutive equation. Together, the nullator and norator are referred to as a *Nullor* [3].

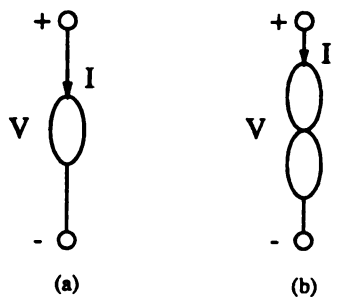


Figure 2.1. (a)Nullator; (b)Norator.

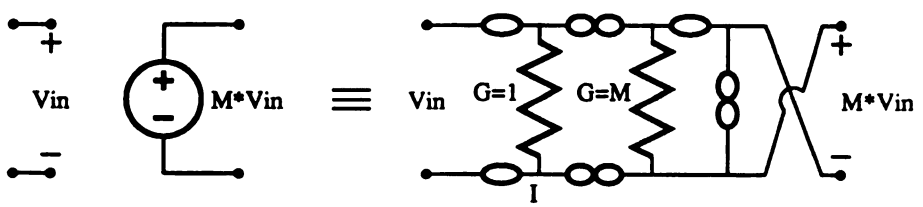


Figure 2.2. Nullator-norator equivalent circuit of a VCVS

All controlled sources, transistors, op-amps, and even inductors can be modeled

using onl

equivalen

been suc

The r

introduc

briefly in

[11] for s

disadva

version

2.2

2.2.1

Writin

The de

and pr

strate

using only resistors, capacitors and nullors [7]. Figure 2.2 shows the nullator-norator equivalent circuit of a voltage controlled voltage source (VCVS). This approach has been successfully implemented in Sspice [4].

The mathematical background for symbolic analog circuit analysis with nullors is introduced in section 2.2. Besides the matrix approach, the graph approach [8] [9] is briefly introduced in section 2.3. Finally, the existing computer programs [5] [4] [10] [11] for symbolic circuit analysis are mentioned in section 2.4 and the advantages and disadvantages of these approaches are discussed, so that the performance of Sspice version 2.0 can be improved.

2.2 Circuits with Nullors

2.2.1 Nullator-Norator Nodal Analysis

Writing nodal network equations by inspection is illustrated in the following example. The details can be found in [3]. The purpose of this example is to take a closer look at and provide a better understanding of circuits with nullors, so that the decomposition strategy of Sspice can be implemented.

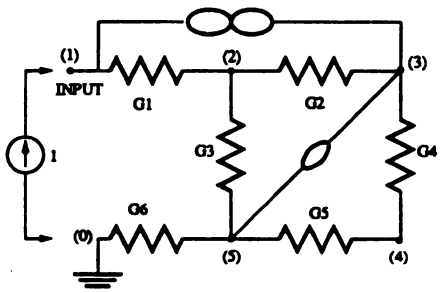


Figure 2.3. A circuit with nullator and norator.

Exam

functi

A

and n

That

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}$$

Ther

betw

colu

prod

Bec

of c

sam

hanc

1 so

Example 3 *Figure 2.3 shows a circuit with a nullator and norator. Find the transfer function of node 4 with respect to node 1.*

Answer: First, we find the nodal equations of the network with the nullator and norator removed and connect a current source with a value of one at the input. That is,

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} G_1 & -G_1 & & & \\ -G_1 & G_1+G_2+G_3 & -G_2 & & -G_3 \\ & -G_2 & G_2+G_4 & -G_4 & \\ & & -G_4 & G_4+G_5 & -G_5 \\ & -G_3 & & -G_5 & G_3+G_5+G_6 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix}. \tag{2.2}$$

Then, each nullator and norator is returned, one by one. Because there is a nullator between node 3 and node 5, V_3 equals V_5. V_3 and V_5 can be combined into $V_{3,5}$ and column 5 of Equation (2.2) is added into column 3, thus, eliminating column 5. This produces

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} G_1 & -G_1 & & \\ -G_1 & G_1+G_2+G_3 & -G_2-G_3 & \\ & -G_2 & G_2+G_4 & -G_4 \\ & & -G_4-G_5 & G_4+G_5 \\ & -G_3 & G_3+G_5+G_6 & -G_5 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_{3,5} \\ V_4 \end{bmatrix}. \tag{2.3}$$

Because of the norator across node 1 and node 3, there would be an arbitrary amount of current flow from node 1 into node 3 or from node 3 into 1. There should be the same amount of current with opposite signs shown at row 3 and row 5 on the left hand side of the above equation. Therefore, row 3 of Equation (2.3) is added into row 1 so that this arbitrary current can be cancelled. Then, row 3 is eliminated, thus,

resulting in the following equation.

$$\begin{bmatrix} I_1+I_3 \\ I_2 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} G_1 & -G_1-G_2 & G_2+G_4 & -G_4 \\ -G_1 & G_1+G_2+G_3 & -G_2-G_3 & \\ & & -G_4-G_5 & G_4+G_5 \\ & -G_3 & G_3+G_5+G_6 & -G_5 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_{3,5} \\ V_4 \end{bmatrix}. \tag{2.4}$$

which is in the form of $\mathbf{I}=\mathbf{Y}\times\mathbf{V}$. Similarly, if there is another nullator (norator) between node i and ground, the V_i (I_i) and its corresponding column (row) is eliminated. Therefore, the node voltages can be obtained by Cramer's rule.

$$\begin{aligned} V_{output} = V_4 &= \frac{det\begin{bmatrix} G_1 & -G_1-G_2 & G_2+G_4 & 1 \\ -G_1 & G_1+G_2+G_3 & -G_2-G_3 & 0 \\ 0 & 0 & -G_4-G_5 & 0 \\ 0 & -G_3 & G_3+G_5+G_6 & 0 \end{bmatrix}}{det(\mathbf{Y})} \\ &= \frac{(-1)^{1+4}\times det\begin{bmatrix} -G_1 & G_1+G_2+G_3 & -G_2-G_3 \\ 0 & 0 & -G_4-G_5 \\ 0 & -G_3 & G_3+G_5+G_6 \end{bmatrix}}{det(\mathbf{Y})} \\ &= \frac{-det(\mathbf{Y}_{1,4})}{det(\mathbf{Y})}. \end{aligned} \tag{2.5}$$

However, $det(\mathbf{Y}_{1,4})$ is the determinant of the admittance matrix of Figure 2.4.

Consequently, the transfer function of Figure 2.3 at node 4 becomes

$$\frac{V_4}{V_1} = \frac{(-1)^{1+4}det(\mathbf{Y}_{1,4})}{(-1)^{1+1}det(\mathbf{Y}_{1,1})}, \tag{2.6}$$

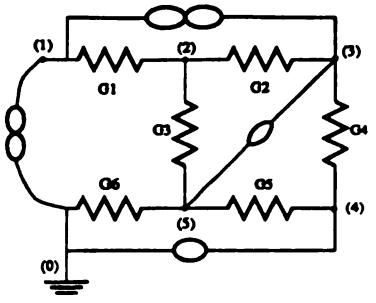


Figure 2.4. Numerator of V_4 in Example 1.

which is the ratio of the network determinants of Figure 2.4 and 2.5 with an appropriate sign. According to the above example, the sign can be obtained by counting the node naming sequence assigned by the user.

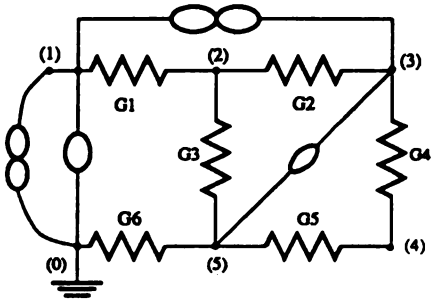


Figure 2.5. The equivalent network determinant of $\mathbf{Y}_{1,1}$.

□

Lemma 1 (Network Determinant, Ndet(N)) *Assume nodes are numbered sequentially with zero for ground. According to the above example, if the column (row) with a higher column (row) number is always merged into the column (row) with a lower number of an admittance matrix when we put nullators (norators) into a circuit, then the determinant of the admittance matrix will be unique for the corresponding*

network with nullors. This network, N, should have a node number assigned to every node. We call this determinant the network determinant of N or Ndet*(N).*

Proof: The admittance matrix of an passive network with all the node numbers assigned is unique. Following the merging rules for nullators and norators stated in Lemma 1 results in a unique admittance matrix if the location of the nullators and norators are decided. □

Definition 1 (Nullator index, u index, u(p,N)) *The u index of node p is the number of nullator trees which connect to nodes whose node number is less than the lowest node number of the nullator tree connecting to node p in network N. A node without any nullator connecting to it is an empty nullator tree.*

Definition 2 (Norator index, o index, o(p,N)) *The o index of node p is the number of norator trees which connect to nodes whose node number is less than the lowest node number of the norator tree connecting to node p in network N. A node without any norator connecting to it is an empty norator tree.*

Definition 1 and Definition 2 describe the *u index* and *o index* which predict the location of the corresponding node in an admittance matrix from the naming of the nodes in a circuit. If the (u index, o index) of a node to which an element G is connected is (j,i), where $j \neq 0$, $i \neq 0$, then a +G would be found at the entry of the ith row and the jth column of the admittance matrix.

Theorem 1 *If the input of a network, N, is node i and ground and the output of the network is node j and ground, then the transfer impedance of the network will be*

$$\frac{(-1)^{m+n}\mathrm{Ndet}(N_1)}{\mathrm{Ndet}(N)}.$$

Here, m (n) is the o (u) index of node i (j), and N_1 is a network with an additional norator between i and ground of N and an additional nullator between j and ground.

Proof: The transfer impedance of the network N can be found by applying a current source of value of one to node i, then the voltage at node j will become the transfer impedance. The network nodal equations are in the following :

$$\begin{aligned}
\mathbf{I} &= \\
\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} &= \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} & \cdots & a_{1,y} \\ \vdots & & \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,n} & \cdots & a_{m,y} \\ \vdots & & \vdots & & \vdots \\ a_{x,1} & \cdots & a_{x,n} & \cdots & a_{x,y} \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ V_j \\ \vdots \\ \vdots \end{bmatrix} \\
&= \mathbf{Y} \times \mathbf{V}.
\end{aligned}$$

The only 1 in the **I** vector will be at the $m'th$ row; and V_j will be at the $n'th$ row of **V**. By using Cramer's rule, the Transfer impedance of network N should be

$$\begin{aligned}
V_j &= \frac{det \begin{bmatrix} a_{1,1} & \cdots & a_{1,n-1} & 0 & a_{1,n+1} & \cdots & a_{1,y} \\ \vdots & & & \vdots & & & \vdots \\ a_{m,1} & \cdots & & 1 & & \cdots & a_{m,y} \\ \vdots & & & \vdots & & & \vdots \\ a_{x,1} & \cdots & & 0 & & \cdots & a_{x,y} \end{bmatrix}}{det(\mathbf{Y})} \\
&= \frac{(-1)^{m+n} \times det(\mathbf{Y}_{m,n})}{det(\mathbf{Y})}.
\end{aligned}$$

$\mathbf{Y}_{m,n}$ is a matrix of **Y** with column n and row m eliminated. Therefore, $\mathbf{Y}_{m,n}$ is the admittance matrix of N with an additional norator grounding a node with an o index

of m, which is node i; and an additional nullator grounding a node with an u index of n, which is node j. □

With the assistance of Theorem 1, the driving point impedance and transfer functions of a network can be obtained by manipulating related network determinants like Equation (2.6).

2.2.2 Equivalence relations

The parallel combination of a nullator with a passive element represents zero voltage across the passive element. Therefore, the current flowing through this passive element would be zero, too, which is equivalent to a nullator only. Using similar analogies, we may conclude the equivalent relations shown in Figure 2.6. According to Figure 2.6, elements may be removed without affecting the network determinant of the circuit. This brings the following definition.

Definition 3 (Prime Network) *If the elimination of any element in a network would change the network determinant of this network, then this network is prime.*

Nullators and norators can be relocated among the nullator and norator trees. Figure 2.7 shows some of the transformations. Furthermore, a circuit with loops of nullators or norators is not prime.

2.2.3 Special Case for Network Determinants

Definition 4 (Invalid Circuit) *A circuit, N, is transformed to be prime. If the number of norators and the number of nullators in this prime circuit are different, then this circuit, N, is called an invalid circuit. Otherwise, it is valid.*

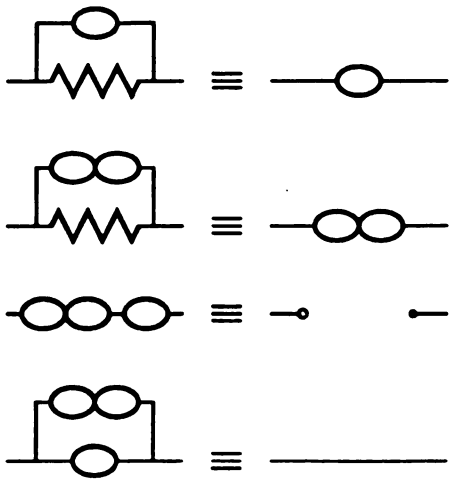


Figure 2.6. Parallel and Open/Short equivalence.

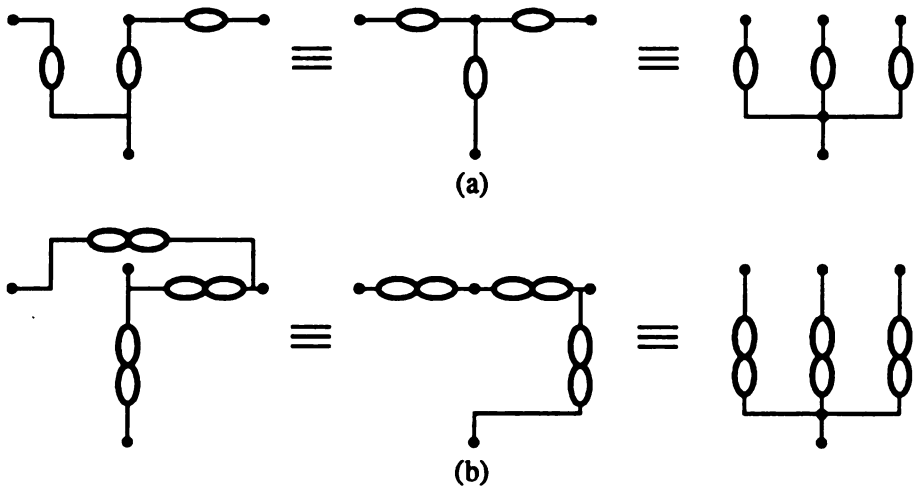


Figure 2.7. (a)Nullator trees; (b)Norator trees.

Lemma 2 *A network can be partitioned into two subnetworks which are connected to each other at only one node. If both of the subnetworks are valid, then the network determinant of this network is the product of the network determinants of the sub-circuits.*

Proof: Without the loss of generality, we assume that a network consists of two parts, say N_1 and N_2, which are connected to each other only at the ground. The admittance matrix, Y, of N can be described as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y_1} & 0 \\ 0 & \mathbf{Y_2} \end{bmatrix}.$$
$$det(\mathbf{Y}) = det(\mathbf{Y_1}) \times det(\mathbf{Y_2}).$$

Because the first co-factors of the indefinite admittance of the network N are all the same, the ground can be selected to be any node of N without changing the network determinant. This concludes the proof. □

According to the above Lemma, if a valid circuit, N, consists of two parts, N_1 and N_2 which are connecting only at a single node as shown in Figure 2.8, then the network determinant of N will be the network determinant of N_1 multiplied by the network determinant of N_2 if both N_1 and N_2 are valid.

However, if the number of nullators and norators in N_1 are not the same, one can easily verify that the network determinant of N will be zero. Therefore, we conclude the following theorem.

Theorem 2 *A valid network can be partitioned into two subnetworks which are connected to each other at only one node. If one of the subnetworks is invalid, then the network determinant of this network is zero.*

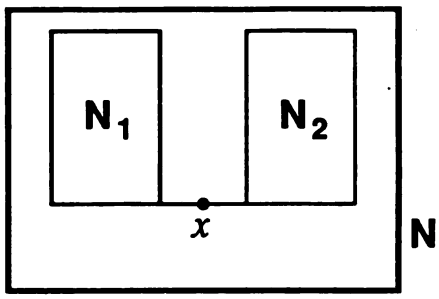


Figure 2.8. A network has two parts connecting at a single node.

Proof: Without the loss of generality, we assume that a network N consists of two parts, say N_1 and N_2, which are connected to each other only at ground. If both N_1 and N_2 are valid circuits individually, then the the admittance matrix, $\mathbf{Y}$, of N can be described as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 & 0 \\ 0 & \mathbf{Y}_2 \end{bmatrix}, where$$

$$\mathbf{Y}_1 = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,m} \end{bmatrix}; and$$

$$\mathbf{Y}_2 = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & & \vdots \\ b_{n,1} & \cdots & b_{n,n} \end{bmatrix}.$$

Now, add a norator into N_1 and a nullator into N_2 so that the first row of $\mathbf{Y}_1$ and the $n'th$ column of $\mathbf{Y}_2$ are eliminated. The new admittance matrix $\mathbf{Y}^+$ becomes

$$\mathbf{Y}^+ = \begin{bmatrix} \mathbf{Y}_1^+ & \mathbf{Y}_3^+ \\ 0 & \mathbf{Y}_2^+ \end{bmatrix};$$

$$\mathbf{Y}_1^+ = \begin{bmatrix} a_{2,1} & \cdots & \cdots & a_{2,m} \\ \vdots & & & \vdots \\ a_{m,1} & \cdots & \cdots & a_{m,m} \\ 0 & \cdots & \cdots & 0 \end{bmatrix};$$

$$\mathbf{Y}_2^+ = \begin{bmatrix} b_{2,1} & \cdots & b_{2,n-1} \\ \vdots & & \vdots \\ b_{n,1} & \cdot & b_{n,n-1} \end{bmatrix};$$

$$\mathbf{Y}_3^+ = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \\ b_{1,1} & \cdots & \cdots & b_{1,n-1} \end{bmatrix}.$$

Since $\mathbf{Y}^+$ is an upper triangular matrix, $det(\mathbf{Y}^+) = det(\mathbf{Y}_1^+) \times det(\mathbf{Y}_2^+)$. Here, $det(\mathbf{Y}_1^+) = 0$. □

In this way, the basic guidelines for decomposing a circuit have been stated. If a circuit can be decomposed into situations in which sub-circuits are connected to each other at only one node, the computation of the network determinant would be less costly. Also, exploring the existence of the invalid sub-circuits can further reduce the computing efforts.

2.3 Graph Theory Approach

The use of graph theory to solve electronic circuit problems is within the realm of nonnumerical algebra for which a considerable amount of literature exists [9] [12] [3] [13]. Unfortunately, most of the methods, although general and powerful, are unduly complicated and are not efficient enough for the needs of analyzing electronic

circuits symbolically. The following subsections briefly introduce two graph methods, signal-flow-graph method and tree-enumeration method.

2.3.1 Coates Graph and Tree-Enumeration Method

In Example 3, the procedure of constructing circuit equations by inspection of active circuits was established. Circuit configurations are designed around a common connection or ground. The selection for ground determines, in many instances, the function or properties of the circuit. An example is the transistor. The common emitter amplifier has the emitter as ground. Likewise, the the common collector amplifier has a grounded collector and common base amplifier has a grounded base. All these circuits are really the same circuit just differing in their common terminal or ground. Suppose that the circuit equations are constructed before a common node is selected, this set of equations must be the same for every circuit of different ground node. This can be illustrated by changing the ground in Figure 2.3 to node number 6. The circuit equations become

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} G_1 & -G_1-G_2 & G_2+G_4 & -G_4 & & \\ -G_1 & G_1+G_2+G_3 & -G_2-G_3 & & & \\ & & -G_4-G_5 & G_4+G_5 & & \\ & -G_3 & G_3+G_5+G_6 & -G_5 & -G_6 & \\ & & -G_6 & & G_6 & \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_{3,5} \\ V_4 \\ V_6 \end{bmatrix}. \quad (2.7)$$

The admittance matrix above is called the indefinite admittance matrix. There are properties which can be observed with the indefinite admittance matrix.

Definition 5 (Indefinite Admittance Matrix) *An indefinite admittance matrix is an admittance matrix in which the sum of the entries in any row or column is zero [9].*

Property 1 *An indefinite admittance matrix is singular [9].*

Property 2 *The determinants of all the first cofactors of an indefinite admittance are the same except the sign [9].*

Mathematically, there are direct relations between a matrix representation and a graph representation. The following is the definition of the Coates graph with respect to the corresponding matrix. Figure 2.9 shows the Coates graph of Equation (2.7).

Definition 6 (Coates Graph) *For a square matrix $A = [a_{ij}]$ of order n, the associated Coates graph is an n-node, weighted, labeled, directed graph, denoted by the symbol $G_c(A)$ or simply G_c if A is clearly understood or is not explicitly given. The nodes are labeled by the integers from 1 to n such that if $a_{ij} \neq 0$, there is an edge directed from node i to node j with associated weight a_{ij} for $i, j = 1, 2, \ldots, n$. Sometimes, it is convenient to consider edges that are not in $G_c(A)$ as edges with zero weight in $G_c(A)$ [9].*

Definition 7 (Associate Coates Graph) *The associate Coates graph of a circuit is the Coates graph of the corresponding indefinite admittance matrix of the circuit [9].*

Theorem 3 *If $\mathbf{Y}$ is an indefinite admittance matrix of a circuit, then the network determinant, $\mathbf{Y}_{ij}$, of the circuit is*

$$\mathbf{Y}_{ij} = \sum_{t_k} f(t_k), \tag{2.8}$$

for $i, j, k = 1, 2, \ldots, n$, where t_k is a directed tree in $G_c(\mathbf{Y})$.

Proof: [9] □

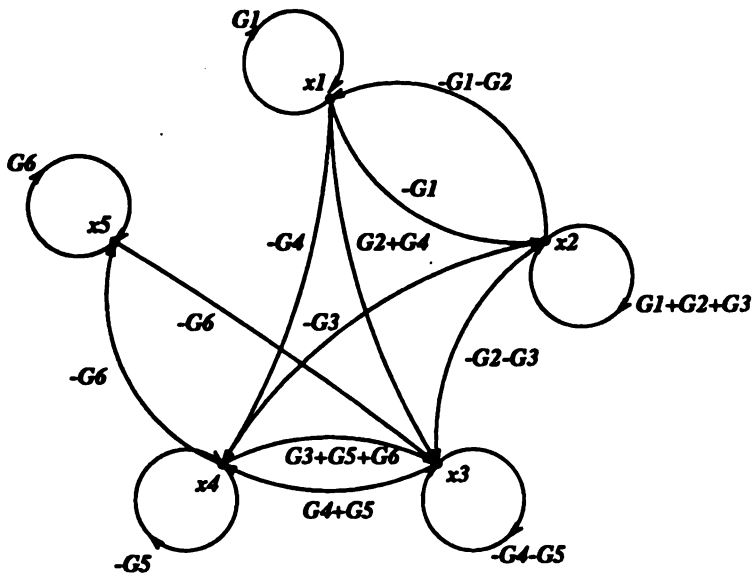


Figure 2.9. A corresponding Coates graph.

According to the above theorem, the network determinant can be obtained by the enumeration of all the directed trees in the associate Coates graph of the circuit. This transforms a circuit problem into a graph algorithm problem, which has been studied and developed extensively. However, advances in graph algorithms cannot keep up the pace of the growth in the size of electronic circuits. The computation efficiency is not acceptable for most practical circuits.

2.3.2 Signal-Flow Graph and Mason's Rule

There is another way to associate a directed graph with a given matrix, known as a signal-flow graph or a Mason graph. The techniques of signal-flow graph have been applied to statistical, mechanical, heat transfer, pneumatic, microwave, and multiple loop feedback systems. In this subsection, a signal-flow graph is called a Mason graph, as distinguished from a Coates graph.

Definition 8 (Mason Graph) *For a given square matrix* **A** *of order* n*, the associated Mason graph of* **A***, denoted by the symbol* $G_m(\mathbf{A})$ *or simply* G_m*, if* **A** *is clearly*

understood or is not explicitly given; is the associated Coates graph of $\mathbf{A} + \mathbf{I}_n$, *where* $\mathbf{I}_n$ *is the identity matrix of order* n *[9].*

The reason for this type of seemingly artificial association is that the Mason graph is a more natural representation of a physical system than the Coates graph [9]. It presents a continual picture of the flow of signals through the physical system, and permits a physical evaluation and a heuristic proof of some basic theorems. Like the Coates graph, the associated Mason graph of a physical system, in many cases, can be drawn directly from the system diagram without the necessity of first setting up the equations in matrix form.

A Mason graph is a weighted, directed graph representing a system of simultaneous linear equations according to the following three rules :

1. Node weights (node variables) represent variables (known or unknown).

2. Branch weights (branch transmittances) represent coefficients in the relationships among node variables.

3. For every node with one or more incoming branches, there corresponds the equation

$$\text{node variable} = \sum(\text{incoming branch transmittance} \times \text{node variable from which the incoming branch originates})$$

where the summation is over all incoming branches (of the node under consideration).

As an example, consider the Mason graph of Figure 2.10. The node variables are x_0, x_1, x_2, x_3, and the branch transmittances are a,b,c,d,e,f,g. According to rule **3**,

this Mason graph represents the following set of equations :

$$\begin{aligned} x_1 &= ax_0 + dx_2, \\ x_2 &= bx_0 + cx_1 + ex_2, \\ x_3 &= fx_1 + gx_2. \end{aligned} \tag{2.9}$$

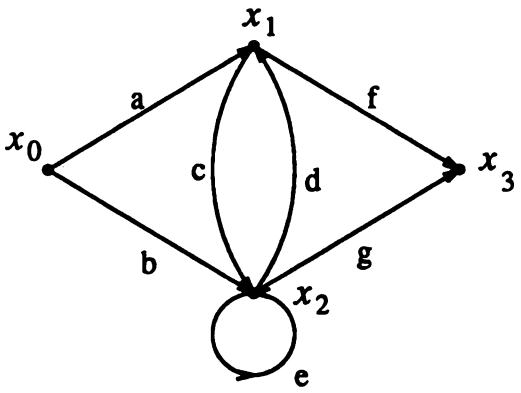


Figure 2.10. A typical Mason graph

In a Mason graph, a node with only outgoing branches is called a *source node.* A node with one or more incoming branches is called a *dependent node.* In particular, a dependent node with only incoming branches is called a *sink node.* For example, in Figure 2.10, x_0 is a source node, x_1, x_2, x_3 are dependent nodes, and x_3 is a sink node. Dependent node variables are treated as unknown quantities and source node variables as known quantities in the simultaneous equations. Therefore, a Mason graph represents a set of simultaneous equations of

$$\mathbf{X}_{n\times 1} = \mathbf{A}_{n\times n}\mathbf{X}_{n\times 1} + \mathbf{B}_{n\times m}\mathbf{Xs}_{m\times 1}. \tag{2.10}$$

These equations can be solved by using Cramer's rule.

A few definitions associated with the Mason graph method are necessary before presenting the topological rule.

Definition 9 (Path.) *Imagine each directed branch in the Mason graph as a one-way street. A path from node* $\mathbf{X}_i$ *to node* $\mathbf{X}_j$ *is any route leaving node* $\mathbf{X}_i$ *and terminating at node* $\mathbf{X}_j$ *along which no node is encountered more than once [9].*

Definition 10 (Loop.) *A loop is a path whose initial node and terminal node coincide [9].*

Definition 11 (*n*th-order Loop.) *An nth-order loop is a set of* n *nontouching loops [9].*

Definition 12 (Path Weight.) *The path weight is the product of all branch transmittances in a path [9].*

Definition 13 (Loop Weight.) *The loop weight is the product of all branch transmittances in a loop [9].*

From Equation (2.10),

$$\mathbf{X} = [\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}\mathbf{X}\mathbf{s}, \tag{2.11}$$

when the inverse of [I-A] exists. Thus, any dependent node variable $\mathbf{X}_j$ may be expressed in terms of the source node variables in the form

$$\mathbf{X}_j = \mathbf{T}_{j1}\mathbf{X}\mathbf{s}_1 + \mathbf{T}_{j2}\mathbf{X}\mathbf{s}_2 + \cdots + \mathbf{T}_{jm}\mathbf{X}\mathbf{s}_m. \tag{2.12}$$

Each $\mathbf{T}_{ji}$ in Equation (2.12) is called the transmission from the source node $\mathbf{Xs}_i$ to the dependent node $\mathbf{X}_j$. The following is a topological rule for evaluating $\mathbf{T}_{ji}$:

$$\mathbf{T}_{ji} = \frac{1}{\Delta}\sum_k P_k\Delta_k, \tag{2.13}$$

where

$$\begin{aligned}
\Delta &= 1 - (\text{sum of all loop weights}) \\
&\quad +(\text{sum of all second order loop weights}) \\
&\quad -(\text{sum of all third order loop weights}) + \cdots \\
P_k &= \text{weight of the } k\text{th path from the source node } \mathbf{Xs}_i \\
&\quad \text{to dependent node } \mathbf{X}_j \\
\Delta_k &= \text{sum of those terms in } \Delta \text{ without any constituent loops} \\
&\quad \text{touching } P_k
\end{aligned} \tag{2.14}$$

and the summation is taken over all paths from $\mathbf{Xs}_i$ to $\mathbf{X}_j$.

According to the above equations, the solutions of the Mason graph can be obtained by finding the nth order loops and paths.

2.4 The Existing Symbolic Circuit Analyzers

The development of algorithms for symbolic analog circuit analysis has proceeded for decades, but, only in recent years has symbolic circuit analysis programs been implemented. Though the internal mechanisms of these programs may differ, using either a matrix based algorithm or a graph based algorithm, their objectives are basically the same. They provide the circuit functions symbolically and further assist analysis with the built-in functions, for example, sensitivity analysis and distortion analysis.

Because of well developed graph algorithms, the first few experimental programs mainly involved symbolic circuit analyzers using graph approaches. SNAP [14] is the most famous one implemented in the early 70s. SC [15] (Symbolic Circuit) is another program of this type.

Because of the inefficiency of graph algorithms for medium sized circuits, recent developments of symbolic circuit analyzers are focused on matrix approaches. Many of the programs are a part of the analog circuit automatic synthesis systems [16] [17] [18]. The emphasis is on the synthesis procedure rather than the symbolic analysis techniques.

Due to the development of symbolic mathematic programs, many symbolic analog circuit analyzers are built on symbolic mathematic packages which are commercially available. For example, SYNAP [19] is built on MACSYMA, and NODAL is built on MATHEMATICA. Both take advantage of the rich graphic and mathematic manipulation capabilities of these packages, so that the systems are better interfaced with the circuit designers. However, the drawback is that the mathematic packages upon which these circuit analyzers are built are general math tools which are not efficient enough for the specific circuit applications. Therefore, they suffer from the low computation speed and high memory consumption which are crucial for real circuit designs.

ISAAC [5] and Sspice [4] are symbolic analog circuit analyzers specialized for the practical circuit designs. Since this dissertation focuses mainly on the details of Sspice, the capabilities of ISAAC are discussed in this section so that the differences between ISAAC and Sspice can be understood.

The success of ISAAC is mainly due to the implementation of a heuristic numerical approximation. Many previously developed programs generate exact expressions only and are difficult for analog designers to analyze, while the simplified circuit expressions are easier for circuit designers to analyze. Besides this capability, ISAAC also provides built-in functions like signal transfer function, internal transfer function, transfer function ratio, rejection ratio, differential mode gain, common mode gain, CMRR, PSRR and many others. These built-in functions are executed by selecting the menu number on the terminal. The only criticism ISAAC may receive is that the functions are fixed and cannot be further manipulated by the circuit designers. In other words,

it is not flexible enough. Also, its efficiency for analyzing large circuits like commercial op-amps and power supply regulators would be questioned since there exists no report on this issue. Therefore, the development of Sspice would focus on flexibility and computation efficiency.

CHAPTER 3

DECOMPOSITION STRATEGIES FOR SYMBOLIC CIRCUIT ANALYSIS

The most common strategies for improving efficiency are circuit decomposition approaches. Of the many methods proposed, the majority are performed at the corresponding mathematical model level, i.e., the graph level, making the procedure difficult to understand and hindering circuit designers from fully utilizing their knowledge and experience. In this chapter, a new circuit level decomposition and composition methodology is described.

3.1 A Review of Decomposition Methods

In order to improve computation efficiency for circuit analysis, decomposition methods have been studied extensively either for numerical or symbolic approaches. For numerical circuit simulations, using relaxation or iterative methods on parallel computers have been a common practice in many computer programs like MSPLICE [20]. It has been found that the effectiveness of the decomposition approach depends on

the connectivity of the circuit to be simulated. The speed-up is usually limited to at most 8 or 9 in some literature reports.

Since the computation efficiency for symbolic circuit analyzers is even worse, decomposition approaches have been studied since the 70s. However, none of the existing decomposition approaches have been suitable for solving real problems for circuit designers. Two criteria need to be satisfied in a successful decomposition method for symbolic circuit analysis. One criterion is an acceptable efficiency. The other is that the output of the circuit analyzer be in an expanded format rather than a sequence of expression format.

According to section 2.3.1, network functions can be obtained by enumerating all the directed trees of an associate Coates graph of a circuit. If the corresponding Coates graph is partitioned into two parts, the directed trees of the Coates graph of the full circuit would be decomposed into multiple trees (k-trees). Therefore, the circuit functions can be calculated by enumerating the needed k-trees, then, combining the appropriate k-trees from different parts to produce the directed trees of the entire circuit [9]. This method has been mathematically well proven. It attracted much attention during the 70s when graph theory and graph algorithms were developed. However, no such circuit analyzer has been really implemented using this method, because enumerating k-trees is not an easy task. The procedure to combine *undirected k-trees* from different parts into an *undirected tree* was developed. But, it is still unclear as to how to combine the *directed k-trees* into *directed trees*, a procedure essential for analyzing general electronic circuits.

The bottom up and top down decomposition of a Mason graph presented in [11] and [21] provide systematic procedures to construct circuit equations. The weaknesses of these approaches are that the circuit should be finely partitioned into small parts and the graph should be loosely connected. Otherwise, the number of different cases to be considered would still be unacceptably large. The method presented in [10]

needs s

The m

expres

3.2

In t

circ

mat

con

Ex

ad

needs symbolic division, which is not suitable for *expanded format* representation. The method based on matrix algorithms like [22] is only suitable for *sequence of expression format* because of the symbolic divisions it needs.

3.2 Obtaining Network Determinants by Decomposition

In this section, the relations among admittance matrices of a circuit and its sub-circuits is discussed. The objective is to find the determinant of the admittance matrix of a circuit from those of its sub-circuits. The following example shows the composition of two sub-circuits and the corresponding admittance matrices.

Example 4 *Figure 3.1 shows that a circuit N consists of two parts,N_a and N_b. The admittance matrices of N_a and N_b are Y_a and Y_b, respectively, where*

$$Y_a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \text{and}$$

$$Y_b = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

Find the admittance matrix for N.

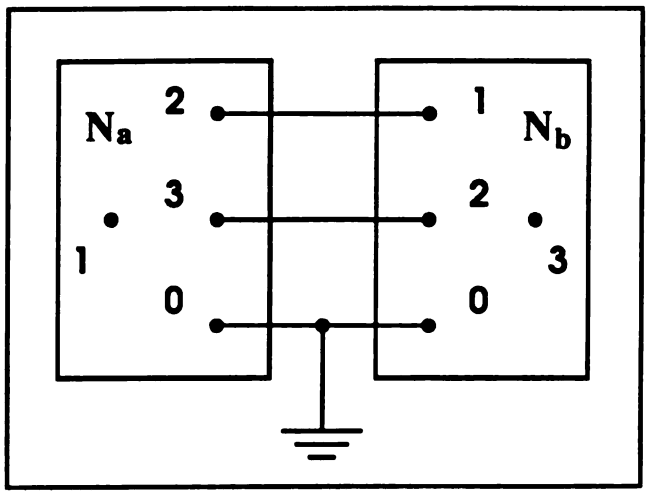


Figure 3.1. Example 2

First, the admittance matrix of N_a and N_b, connected to each other only at ground, is written as

$$Y^{++} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & & & \\ & & & b_{11} & b_{12} & b_{13} \\ & & & b_{21} & b_{22} & b_{23} \\ & & & b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

Then, short node 2 of N_a with node 1 of N_b, as described in Figure 3.1, which is the same as putting a nullator-norator pair between these two nodes. Therefore, the fourth column and the fourth row of Y^{++} are added into the second column and row;

then, the fourth column and row of Y^{++} are removed. This results in

$$Y^{+} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & \\ a_{21} & a_{22}+b_{11} & a_{23} & b_{12} & b_{13} \\ a_{31} & a_{32} & a_{33} & & \\ & b_{21} & & b_{22} & b_{23} \\ & b_{31} & & b_{32} & b_{33} \end{bmatrix}.$$

Again, node 3 of N_a and node 2 of N_b are shorted by adding the fourth column and row of Y^{+} into the third column and row; then, the fourth column and row are removed. As a result, the admittance matrix of N becomes

$$Y = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \\ a_{21} & a_{22}+b_{11} & a_{23}+b_{12} & b_{13} \\ a_{31} & a_{32}+b_{21} & a_{33}+b_{22} & b_{23} \\ & b_{31} & b_{32} & b_{33} \end{bmatrix}. \tag{3.1}$$

□

Based on this example, the relations among Y_a, Y_b, and $det(Y)$ are explored by reversing the matrix construction process. Expanding Equation (3.1) at the second column yields

$$\begin{aligned} det(Y) &= -a_{12} \times Y_{1,2} + (a_{22}+b_{11}) \times Y_{2,2} - (a_{32}+b_{21}) \times Y_{3,2} + b_{31} \times Y_{4,2} \\ &= -a_{12} \times Y_{1,2} + a_{22} \times Y_{2,2} - a_{32} \times Y_{3,2} + b_{31} \times Y_{4,2} \\ &\quad -0 \times Y_{1,2} + b_{11} \times Y_{2,2} - b_{21} \times Y_{3,2} + b_{31} \times Y_{4,2} \end{aligned}$$

$$= det\begin{bmatrix} a_{11} & a_{12} & a_{13} & \\ a_{21} & a_{22} & a_{23}+b_{12} & b_{13} \\ a_{31} & a_{32} & a_{33}+b_{22} & b_{23} \\ & & b_{32} & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & & a_{13} & \\ a_{21} & b_{11} & a_{23}+b_{12} & b_{13} \\ a_{31} & b_{21} & a_{33}+b_{22} & b_{23} \\ & b_{31} & b_{32} & b_{33} \end{bmatrix} \quad (3.2)$$

Using the same method, the decomposed matrices are further decomposed at the third column and the second and third rows, iteratively, as follows :

$$det(Y) = \quad (3.3)$$

$$det\begin{bmatrix} a_{11} & a_{12} & a_{13} & \\ a_{21} & a_{22} & a_{23} & \\ a_{31} & a_{32} & a_{33} & \\ & & & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & & \\ & & b_{22} & b_{23} \\ & & b_{32} & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & a_{12} & & \\ & & b_{12} & b_{13} \\ a_{31} & a_{32} & & \\ & & b_{32} & b_{33} \end{bmatrix}$$

$$+det\begin{bmatrix} a_{11} & & a_{13} & \\ a_{21} & & a_{23} & \\ & b_{21} & & b_{23} \\ & b_{31} & & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & & a_{13} & \\ & b_{11} & & b_{13} \\ a_{31} & & a_{33} & \\ & b_{31} & & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & & & \\ & b_{11} & b_{12} & b_{13} \\ & b_{21} & b_{22} & b_{23} \\ & b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

Then, as the a entries and b entries are collected together,

$$det(Y) = \quad (3.4)$$

$$det\begin{bmatrix} a_{11} & a_{12} & a_{13} & \\ a_{21} & a_{22} & a_{23} & \\ a_{31} & a_{32} & a_{33} & \\ & & & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & & \\ & & b_{22} & b_{23} \\ & & b_{32} & b_{33} \end{bmatrix} - det\begin{bmatrix} a_{11} & a_{12} & & \\ a_{31} & a_{32} & & \\ & & b_{12} & b_{13} \\ & & b_{32} & b_{33} \end{bmatrix}$$

$$-det\begin{bmatrix} a_{11} & a_{13} & & \\ a_{21} & a_{23} & & \\ & & b_{21} & b_{23} \\ & & b_{31} & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & a_{13} & & \\ a_{31} & a_{33} & & \\ & & b_{11} & b_{13} \\ & & b_{31} & b_{33} \end{bmatrix} + det\begin{bmatrix} a_{11} & & & \\ & b_{11} & b_{12} & b_{13} \\ & b_{21} & b_{22} & b_{23} \\ & b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

According to Equation (3.5), all the decomposed matrices are block diagonal matrices. Each block consists of all a's or all b's and belongs to a specific sub-circuit. These blocks are variations of Y_a or Y_b obtained by eliminating columns or rows and are modeled by putting nullators and norators at the corresponding nodes. Decomposition information, therefore, can be provided at circuit level. Figure 3.2 shows the computation in a more illustrative way and corresponds to the six cases in Equation (3.5).

3.3 The Algorithm

The following is the algorithm which obtains the network determinant of a circuit by decomposition as shown in Example 4.

```
ALGORITHM :

Ndet_by_Decomposition(N)
/* N is the network */
/* N is decomposable into Na and Nb */
/* The node numbers in Na are all less than the node numbers in Nb */
{
Result=NULL;
P<-Identify_cut_nodes(N); /* Get the cut nodes except the GND */
Na,Nb<-Decompose(N,P);
#_of_case=2**(#_of_P);
For(i=0;i<#_of_case;i++) {
     For(j=0;j<#_of_case;j++) {
```

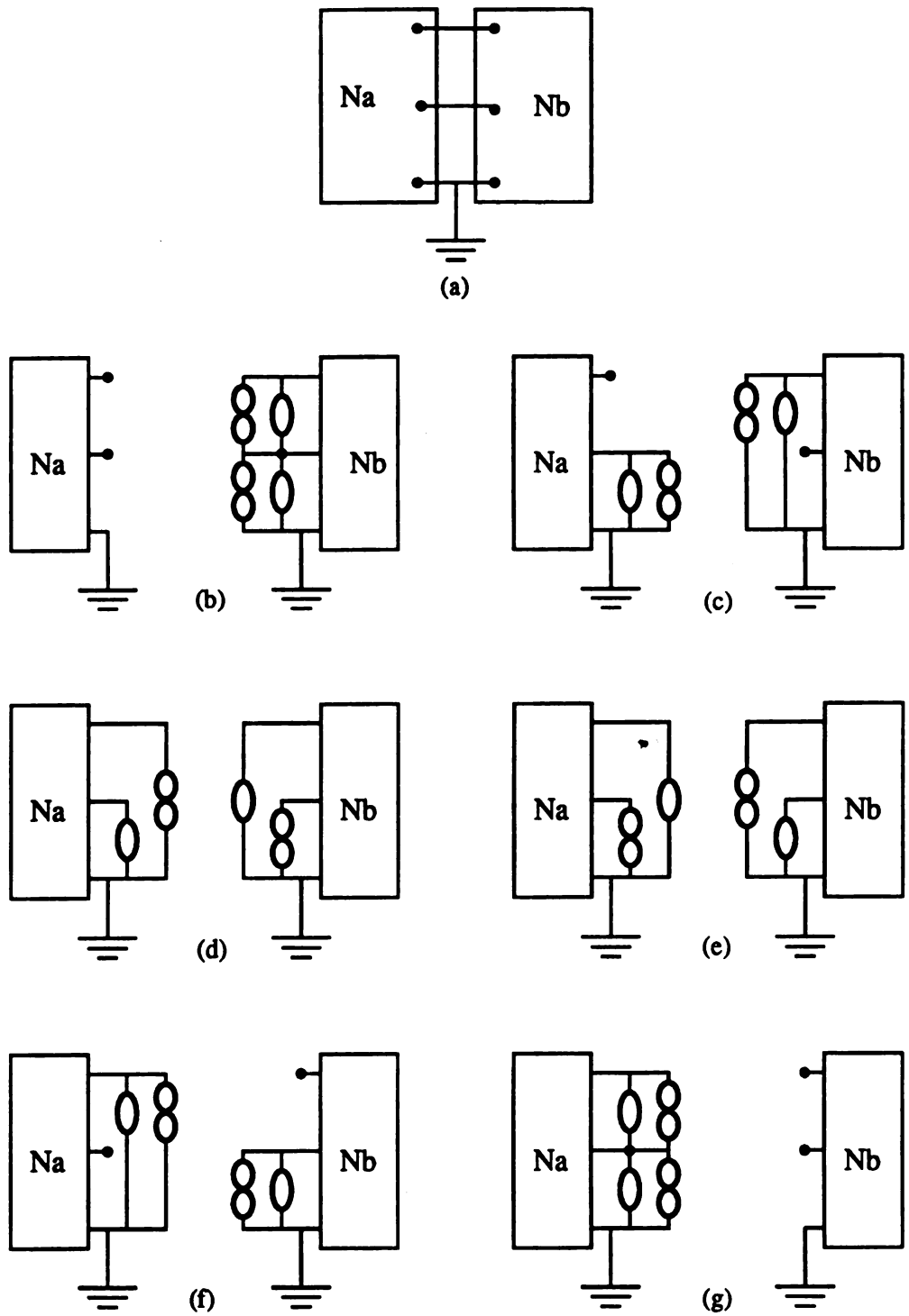


Figure 3.2. The decomposition and composition of two sub-circuits connected at three nodes; (a) The whole circuit; (b)-(g) six cases.

```
        Pu=encode_decimal_to_binary(i,#_of_P);
             /* when i==2 and #_of_P==4, Pu='0010' */
        Po=encode_decimal_to_binary(j,#_of_P);
        For(m=0;m<#_of_P;m++) {
             If(Pu[m]==1} {
                  Put_nullator_between_P[m]_and_GND_in_Na;
                  }
             else {
                  Put_nullator_between_P[m]_and_GND_in_Nb;
                  }
             If(Po[m]==1} {
                  Put_norator_between_P[m]_and_GND_in_Na;
                  }
             else {
                  Put_norator_between_P[m]_and_GND_in_Nb;
                  }
             }
        If( Valid(Na) && Valid(Nb) ) {
             A=Ndet(Na);
             B=Ndet(Nb);
             Result=Result+A*B*Sign(N,Na,Nb);
             }
        Remove_the_added_nullator_and_norator_at_P;
        }
    }
Return(Result);
}
```

Suppose the sub-circuits, **Na** and **Nb**, are connected at n nodes, except the ground, the above algorithm would connect n nullators and n norators between each connection node and the **GND**, either in **Na** or in **Nb**, for all the possible combinations. Each combination is a decomposition case. Then, the algorithm needs to examine whether the newly formed sub-circuits are valid for each decomposition case. If both of them are valid, their network determinants can be obtained independently by any of the existing network determinant algorithms without decomposition. The network

determinants obtained are, then, multiplied together with an appropriate sign. In Example 4, there are only six valid decomposition cases. The validity of a circuit and the procedure to obtain the signs are defined next.

Example 5 *Figure 2.4 shows a circuit. What are the u and o indexes of node (4)?*

Answer: There are four nodes whose node numbers are less than 4. They are (0), (1), (2), and (3). Both node 0 and node 2 have no norators connected to them and are counted into two norator trees which are empty. Also, node 1 and 3 belong to the same norator tree. Therefore, the o index of node 4 is three. The u index of node 4 can be found in the same way and it is four. □

Definition 14 (Inverse u nodes, iu nodes, iu(k,N)) *iu(k,N) represents all the nodes whose u index is k in the network of N.*

Definition 15 (Inverse o nodes, io nodes, io(k,N)) *io(k,N) represents all the nodes whose o index is k in the network of N.*

Actually, the u index and the o index represent the location of a node in an admittance matrix. If there is an element G connected to a node whose (u index, o index) is (j,i) and $i \neq 0, j \neq 0$, then there will be a term of +G at the entry of i'th row and j'th column in the corresponding admittance matrix.

Because the order of columns and rows in an admittance matrix can affect the sign of its determinant, as shown in equation (3.5), this can be modeled by the permutations of the u and o indexes.

Definition 16 (Decomposed u index, du index, du(p,N,case)) *The du index of node p is the u index of node p in N with nullators and norators added under a specific decomposition case. N can be decomposed into* Na *and* Nb. *The node numbers of the nodes in* Na *are smaller than those in* Nb.

Definition 17 (Inverse du nodes, idu(k,N,case)) *idu(k,N,case) represents the set of nodes whose du indexes are k for the specific decomposition case of network N.*

Definition 18 (Decomposed o index, do index, do(p,N,case)) *The do index of node p is the o index of node p in N with nullators and norators added under a specific decomposition case. N can be decomposed into* **Na** *and* **Nb**. *The node numbers of the nodes in* **Na** *are smaller than those in* **Nb**.

Definition 19 (Inverse do nodes, ido(k,N,case)) *ido(k,N,case) represents the set of nodes whose do indexes are k for the specific decomposition case of network N.*

Definition 20 (Connected du index, cdu index, cdu(k,N,case))
The connected du index is the u index of the nodes whose du index is k for a specific decomposition case in network N before the decomposition scheme is applied. cdu(k,N,case)=u(idu(k,N,case),N).

Definition 21 (Connected do index, cdo index, cdo(k,N,case))
The connected do index is the o index of the nodes whose do index is k for a specific decomposition case in network N before the decomposition scheme is applied. cdo(k,N,case)=o(ido(k,N,case),N).

Example 6 *Figure 3.3(a) shows a circuit that consists of two parts,* **Na** *and* **Nb**. *By applying the above decomposition algorithm, a* **case**, *as shown in Figure 3.3(b), simular to Figure 3.2(d) results. Find the du and do indexes of nodes (1') and (2') for this* **case**. *Assume all the node numbers in* **Nb** *to be larger than those in* **Na**. *Also, find the* **cdu(du(2',N,case),N,case)** *and* **cdo(do(1',N,case),N,case)**.

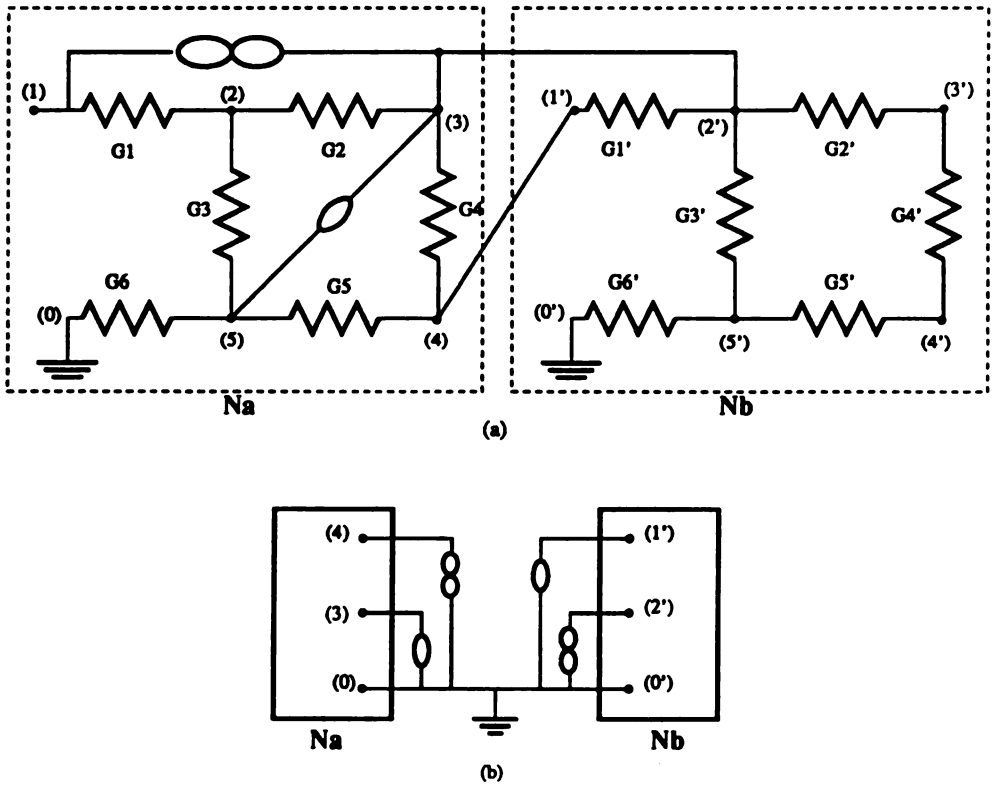


Figure 3.3. Example with Na and Nb

The (du index, do index) for (1') in Figure 3.3(b) is (0,5) and for (2') is (6,0). `cdo(do(1',N,case),N,case)` is 3 and `cdu(du(2',N,case),N,case)` is also 3. □

According to the above definitions, each du or do index of a specific decomposition case has a cdu or cdo index which describes the relocation of the column or the row in the admittance matrix before the decomposition is conducted. This is referred to in section 3.2. When only one of the du to cdu permutation and the do to cdo permutation is odd, a -1 should be multiplied to this decomposition case. We may conclude the following algorithms.

ALGORITHM :

```
Valid(N)
{
u=max(all_the_possible_u_index(N));
o=max(all_the_possible_o_index(N));
if(u==o) {
     return(1);
     }
else {
     return(0);
     }
}

Sign(N,Na,Nb)
{
/* U is an array with the length of the maximum number of u index in N */
/* O is an array with the length of the maximum number of o index in N */
/* The length of U and the length of O should be the same */
k=max(all_the_possible_u_index(N));
For(i=0;i<k;i++) {
     U[i]=cdu(i,N,case(Na,Nb));
     O[i]=cdo(i,N,case(Na,Nb));
     }
If( (odd_permutation(U) && odd_permutation(O)) ||
    (even_permutation(U) && even_permutation(O)) ) {
```

```
        return(1);
        }
else {
        return(-1);
        }
}
```

3.3.1 Analysis

The next question is whether decomposition is universally beneficial for symbolic circuit analysis under all situations. In this subsection, a simple analysis is presented.

Suppose there is a network of dimension N and average connectivity q. If the DD algorithm [23] for obtaining matrix determinant is used, the complexity of this computation is

$$F_{DD}(N,q) \leq q^{N-q} \times q!. \tag{3.5}$$

Now, let this network be partitioned into two parts whose dimensions are assumed to be $N/2$. The number of cut nodes except for the ground is m. If both parts are valid circuits, then using the decomposition algorithm in section 3.3 yields

$$F_{Case}(m) = \sum_{i=0}^{m} (C_m^i)^2 = \sum_{i=0}^{m} \left(\frac{m!}{(m-i)!i!}\right)^2$$

valid cases. Example 4 is an example of $m = 2$. The computational complexity with one level of decomposition becomes

$$F_{Decompose,DD}(N,q,m) = F_{DD}(N/2,q) \times 2 \times F_{Case}(m). \tag{3.6}$$

Here, some assumptions are made. First, the average connectivity remains the same before and after the network is decomposed. Second, the dimension of each decomposed case is the same. This is not true, according to Equation (3.5), where the

dimensions of the decomposed sub-matrices range from 1 to 4. Since the purpose of this equation is to give a rough estimate, for simplicity's sake, we can accept these assumptions.

The ratio between Equation (3.5) and Equation (3.6) shows whether it is beneficial to do circuit decomposition.

$$\begin{aligned} F_{Evaluate,DD}(N,q,m) &\leq log(\frac{F_{DD}(N,q)}{F_{Decompose,DD}(N,q,m)}) \\ &= log(\frac{q^{N/2}}{2\sum_{i=0}^{m}(\frac{m!}{(m-i)!i!})^2}). \end{aligned} \tag{3.7}$$

If the circuit dimension N and the average connectivity of this circuit q are constant, the maximum number of m which makes the $F_{Evaluate,DD}$ positive represents the maximum number of cut nodes beneficial to decomposition. For example, $F_{Evaluate,DD}(N,q,2) > 0$ while $F_{Evaluate,DD}(N,q,3) < 0$ means that if the circuit can be partitioned at two nodes, the application of decomposition improves the computation efficiency. If this circuit can only be partitioned at three nodes, then decomposition is not beneficial to this circuit.

If the determinant obtaining algorithm is improved, then the computational complexity becomes

$$F_{Improve}(N,q,f) \leq \frac{q^{N-q} \times q!}{f(N)}. \tag{3.8}$$

Similar to Equation (3.7), this results in

$$\begin{aligned} F_{Evaluate,Improve}(N,q,m,f) &\leq log(\frac{F_{Improve}(N,q,f)}{2 \times F_{Improve}(N/2,q) \times F_{Case}(m)}) \\ &= log(\frac{q^{N/2} \times f(N/2)}{2 \times f(N) \times \sum_{i=0}^{m}(\frac{m!}{(m-i)!i!})^2}). \end{aligned} \tag{3.9}$$

When $f(N) = 1$, Equation (3.9) is equal to Equation (3.7). Figure 3.4 shows the

trend of maximum number of cut nodes, m, that keep $F_{Evaluate,Improve} > 0$ for different circuit dimensions, N, while the average connectivity, q, is 3.5.

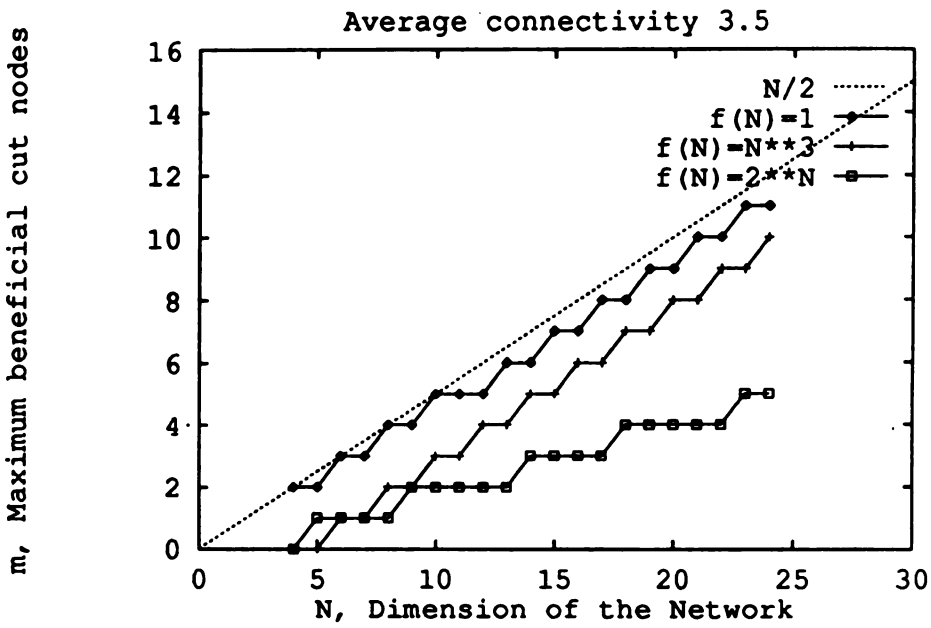


Figure 3.4. Maximum beneficial cut nodes analysis.

According to Figure 3.4, as the dimension of the circuit increases, the acceptable number of cut nodes increases. Also, as the computation complexity of the network determinant obtaining algorithm improves, the acceptable number of cut nodes decreases. In [23], it shows that the complexity of the resistor chain with length n is $O(2^n)$ when the DD algorithm is used. On the other hand, the complexity for the SLE/M algorithm [23] is $O(n^2)$. Similarly, according to Equation 3.9), one can easily find that decomposition becomes not so attractive for loosely connected circuits. Therefore, the application of decomposition to an ordinary circuit with a dimension less than 8 is not encouraged while using *Sspice version 2.0.*

3.4 Exploiting Special Decomposition Opportunities

According to the procedure described in Equation (3.2), it should show 16 decomposed cases in Figure 3.2; however, 10 of them consist of invalid sub-circuits. Their network determinants are identified to be zero, according to Theorem 2. Each case in Figure 3.2 is a circuit consisting of two valid sub-circuits connected to each other only at the ground. According to Lemma 1, the network determinants can be obtained from those of the sub-circuits.

The above is a generalized procedure. For some situations, special rules may be very usefully to simplify the procedure.

Theorem 4 *A network N consists of two parts, N_1 and N_2, where N_1 and N_2 are valid circuit. If they are connected at two points, at the ground and at node x, then the network determinant of N is*

$$(-1)^{p+k} \times \mathrm{Ndet}(N_1) \times \mathrm{Ndet}(\hat{N}_2) + (-1)^{q+k} \times \mathrm{Ndet}(\hat{N}_1) \times \mathrm{Ndet}(N_2).$$

Here, $\hat{N}_1$ ($\hat{N}_2$) is N_1 (N_2) with node x shorted to ground. p, q and k are the summations of the o and u indexes of node x in N_1, N_2 and N, respectively.

Proof: The input admittance of N at x is equal to

$$\begin{aligned}
& \frac{Ndet(N)}{(-1)^k \times Ndet(\hat{N})} \quad (Theorem\ 1) \\
= & \frac{Ndet(N_1)}{(-1)^p \times Ndet(\hat{N}_1)} + \frac{Ndet(N_2)}{(-1)^q \times Ndet(\hat{N}_2)} \quad (Theorem\ 1) \\
= & \frac{(-1)^p \times Ndet(N_1)}{Ndet(\hat{N}_1)} + \frac{(-1)^q \times Ndet(N_2)}{Ndet(\hat{N}_2)} \\
= & \frac{(-1)^p \times Ndet(N_1) \times Ndet(\hat{N}_2) + (-1)^q \times Ndet(\hat{N}_1) \times Ndet(N_2)}{Ndet(\hat{N}_1) \times Ndet(\hat{N}_2)},
\end{aligned}$$

where $\hat{N}$ is N with node x shorted to ground. Because $Ndet(\hat{N})$ is equal to $Ndet(\hat{N}_1) \times Ndet(\hat{N}_2)$, this theorem is concluded. □

Similar to the theorem of Parameter Extraction [13], the network determinant can be decomposed into two parts, one consisting of terms with a specific element, say G, and the other consisting of terms without this element. It becomes a special case of Theorem 4 when N_2 is a single element, and may result in the following corollary.

Corollary 1 *If there exists an element G connecting nodes p and q in network N, then the network determinant of N is equal to* $Ndet(N_1)+(-1)^{k+l+m} \times G \times Ndet(N_2)$. *Here, k (l) is the largest o (u) index of p and q. When both o and u indexes of p are larger or smaller than those of q, m=0. Otherwise, m=1.* N_1 *is a network of N without G;* N_2 *is a network of N with an additional pair of nullator and norator parallel with G. If either o or u indexes of p and q are equal, then* $Ndet(N_2)=0$.

Example 7 *According to Corollary 1, the network determinant of Figure 3.5(a) is equal to* $Ndet(N_1)+(-1) \times \mathtt{G4} \times Ndet(N_2)$. N_1 *is shown in Figure 3.5(b), and* N_2 *is shown in Figure 3.5(c).* $\mathrm{Ndet}(N_1)=0$ *by Theorem 2.*

Theorem 5 *If there exists a nullator (norator) connecting nodes p and q in network N, the network determinant,* $\mathrm{Ndet}(N)$, *is equal to* $Ndet(N_1)+(-1)^{|j-l|-1} \times Ndet(N_2)$. *Here, j is the u (o) index of node p and l is the u (o) index of q in the circuit of* N^+. N_1 *is a circuit which relocates the nullator (norator) connecting p and q in N to p or q with a larger u (o) index in* N^+ *and ground.* N_2 *is a circuit which relocates the nullator (norator) connecting p and q in N to p or q with a smaller u (o) index in* N^+ *and ground.* N^+ *is N without the nullator (norator) connecting p and q. When j=l, this nullator (norator) is redundant.*

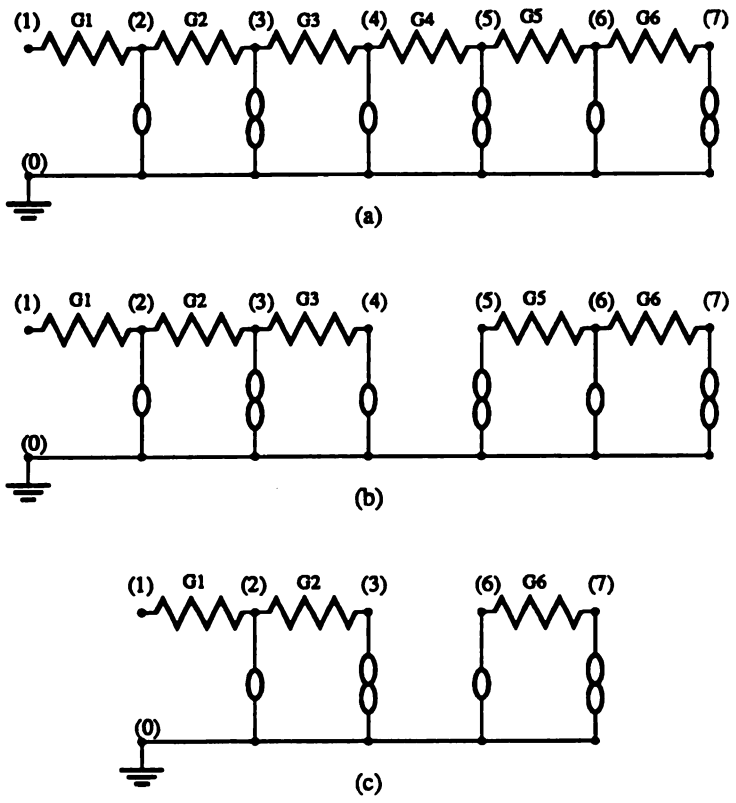


Figure 3.5. Example 3.

Proof: For the nullator, since the admittance matrix of N^+ can be laid out as

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,j} & \cdots & a_{1,l} & \cdots & a_{1,n+1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j} & \cdots & a_{n,l} & \cdots & a_{n,n+1} \end{bmatrix},$$

which is an $n \times (n+1)$ matrix, the admittance matrix of N_1 becomes

$$\mathbf{Y}_1 = \begin{bmatrix} a_{1,1} & \cdots & a_{1,j} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j} & \cdots & a_{n,l-1} & a_{n,l+1} & \cdots & a_{n,n+1} \end{bmatrix}$$

if $l > j$; and

$$\mathbf{Y}_2 = \begin{bmatrix} a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l} & \cdots & a_{1,n+1} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j-1} & a_{n,j+1} & \cdots & a_{n,l} & \cdots & a_{n,n+1} \end{bmatrix}.$$

Both $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are $n \times n$ matrixes. Also, the admittance matrix of N should be

$$\mathbf{Y} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,j} + a_{1,l} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j} + a_{n,l} & \cdots & a_{n,l-1} & a_{n,l+1} & \cdots & a_{n,n+1} \end{bmatrix}.$$

$$det(\mathbf{Y}) = (-1)^{j-1} \times$$

$$det \begin{bmatrix} a_{1,j}+a_{1,l} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ a_{1,j}+a_{1,l} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \end{bmatrix}$$

$$= (-1)^{j-1} \times det(\mathbf{Y}^{+})$$

$$= (-1)^{j-1} \times \sum_{i=1}^{n} (a_{i,j} + a_{i,l}) \times \mathbf{Y}^{+}_{i,1}.$$

$$det(\mathbf{Y}_1) = (-1)^{j-1} \times$$

$$det \begin{bmatrix} a_{1,j} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ a_{1,j} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \end{bmatrix}$$

$$= (-1)^{j-1} \times det(\mathbf{Y}^{++})$$

$$= (-1)^{j-1} \times \sum_{i=1}^{n} (a_{i,j}) \times \mathbf{Y}^{++}_{i,1}.$$

$$det(\mathbf{Y}_2) = (-1)^{l-1-1} \times$$

$$det \begin{bmatrix} a_{1,l} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdots & a_{1,n+1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ a_{1,l} & a_{1,1} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1,l-1} & a_{1,l+1} & \cdot & a_{1,n+1} \end{bmatrix}$$

$$= (-1)^{l-2} \times det(\mathbf{Y}^{+++})$$

$$= (-1)^{l-2} \times \sum_{i=1}^{n} (a_{i,l}) \times \mathbf{Y}^{+++}_{i,1}.$$

Because $\mathbf{Y}^{+}_{i,1} = \mathbf{Y}^{++}_{i,1} = \mathbf{Y}^{+++}_{i,1}$,

$$
\begin{aligned}
det(\mathbf{Y}_1) &= (-1)^{j-1} \times \sum_{i=1}^{n}(a_{i,j}) \times \mathbf{Y}^{+}_{i,1}; \\
det(\mathbf{Y}_2) &= (-1)^{l-2} \times \sum_{i=1}^{n}(a_{i,l}) \times \mathbf{Y}^{+}_{i,1}.
\end{aligned}
$$

$$
\begin{aligned}
det(\mathbf{Y}) &= (-1)^{j-1} \sum_{i=1}^{n} a_{i,j}\mathbf{Y}^{+}_{i,1} + (-1)^{j-1} \sum_{i=1}^{n} a_{i,l}\mathbf{Y}^{+}_{i,1} \\
&= det(\mathbf{Y}_1) + (-1)^{j-1} \times (-1)^{(l-2)-(j-1)} \times (-1)^{(l-2)-(j-1)} \times \sum_{i=1}^{n} a_{i,l}\mathbf{Y}^{+}_{i,1} \\
&= det(\mathbf{Y}_1) + (-1)^{l-j-1} \times (-1)^{(l-2)} \times \sum_{i=1}^{n} a_{i,l}\mathbf{Y}^{+}_{i,1} \\
&= det(\mathbf{Y}_1) + (-1)^{l-j-1} \times det(\mathbf{Y}_2).
\end{aligned}
$$

When $j > l$, $det(\mathbf{Y}) = det(\mathbf{Y}_1) + (-1)^{j-l-1} \times det(\mathbf{Y}_2)$. This concludes the nullator part of the theorem.

Similarly, the norator part of the proof can be done in the same way. □

Example 8 *According to Theorem 5, the network determinant of Figure 3.6(a) is equal to the summation of the network determinant of Figure 3.6(b) and Figure 3.6(c). Table 3.1 shows the verification.*

Figure	Network Determinant
Figure 8(a)	`-G3*G1`
Figure 8(b)	`-G3*G1-G2*G1`
Figure 8(c)	`+G2*G1`

Table 3.1. Symbolic Network Determinants of Example 4.

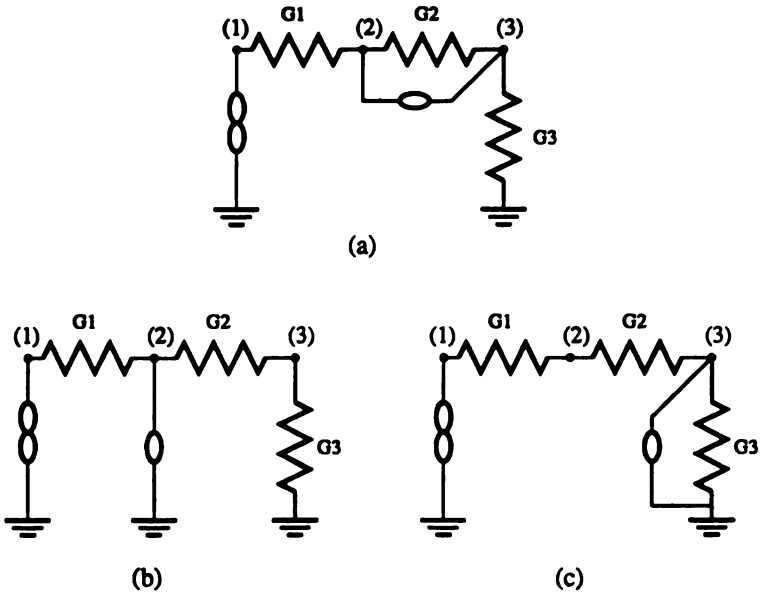


Figure 3.6. Example 4.

3.5 A Circuit Level Decomposition Application

Figure 3.7 shows a circuit which consists of two parts, connected at 4 nodes, including the ground. $\mathbf{A}_{dm}$ is a voltage controlled voltage source with an undefined transfer function. Supposing both $\mathbf{A}_{dm}$ and **B** are complicated circuits, the process for computing the symbolic network determinant of this circuit, ND, would need decomposition so that the transfer function can be obtained efficiently. However, according to the methodology described in Equation (3.2)-(3.5) and Figure 3.2, it may produce 20 different cases. In this section, we will show how Theorem 4 and Theorem 5 further simplify the proposed circuit level decomposition strategy.

Without affecting the circuit characteristics, two voltage controlled voltage sources with values of one are inserted between nodes A, B, D, E, and nodes C, F, ground. The voltage controlled voltage sources are substituted by the norator-nullator equivalent circuit as shown in Figure 2.2. This may produce Figure 3.8. In this way, we can

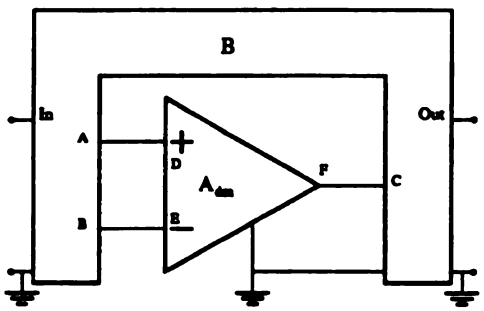


Figure 3.7. An Example Circuit with two parts.

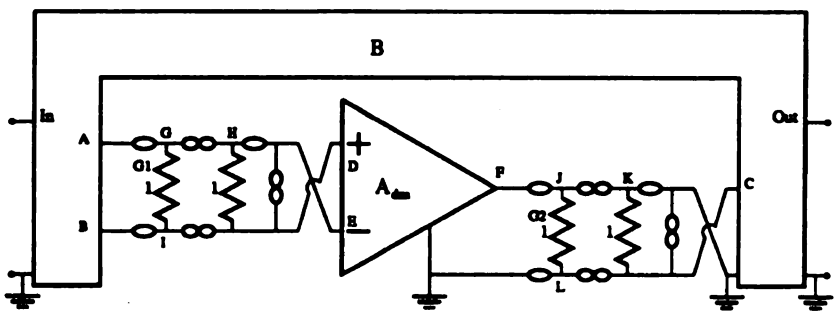


Figure 3.8. The Equivalent Circuit

Nodes	(*o index* , *u index*)
A	(a_o, a_u)
B	(b_o, b_u)
C	(c_o, c_u)
D	(d_o, d_u)
E	(e_o, e_u)
F	(f_o, f_u)
G	(g_o, g_u)=(g_o, a_u)
H	(h_o, h_u)=(g_o, e_u)
I	(i_o, i_u)=(d_o, b_u)
J	(j_o, j_u)=(j_o, f_u)
K	(k_o, k_u)=$(j_o, 0)$
L	(l_o, l_u)=$(c_o, 0)$

Table 3.2. *o* and *u* indexes.

perform the decomposition at the inserted circuits so that the original circuits, $\mathbf{A}_{dm}$ and **B**, are kept unchanged.

B is an m by m circuit and $\mathbf{A}_{dm}$ is n by n. The inserted circuits produce extra nodes to Figure 3.8. All these nodes have their unique node numbers and L > K > J > I > H > G > AllOtherNodes. The *o index* and *u index* of each node can be listed as shown in Table 3.2.

We assume that $b_o > a_o$, $b_u > a_u$, $d_o > e_o$, and $d_u > e_u$. According to Corollary 1, the network determinant of Figure 3.8 is equal to

$$\texttt{ND1a} + (-1)^{g_o+b_u+1} \times \texttt{ND2}, \tag{3.10}$$

where **ND1a** and **ND2** are the network determinants of Figure 3.9(a) and Figure 3.9(c), respectively. This equation can be further simplified to

$$(-1)^{g_o+d_u+1} \times \texttt{ND1} + (-1)^{g_o+b_u+1} \times \texttt{ND2}, \tag{3.11}$$

where ND1 is the network determinant of Figure 3.9(b).

Again, by applying Corollary 1, ND1 is found to be

$$\begin{aligned} & \mathtt{ND11a} + (-1)^{j_o+f_u} \times \mathtt{ND12} \\ = \; & (-1)^{j_o+c_u+1} \times \mathtt{ND11} + (-1)^{j_o+f_u} \times \mathtt{ND12}, \end{aligned} \tag{3.12}$$

where ND11a, ND11, and ND12 are network determinants of those circuits shown in Figure 3.10. Because the circuit in Figure 3.10(c) has invalid sub-circuits,

$$\mathtt{ND1} = (-1)^{j_o+c_u+1} \times \mathtt{ND11}. \tag{3.13}$$

The same procedure can be applied to ND2. Therefore,

$$\begin{aligned} \mathtt{ND2} &= \mathtt{ND21} + (-1)^{j_o+f_u} \times \mathtt{ND22} \\ &= (-1)^{j_o+f_u} \times \mathtt{ND22}. \end{aligned} \tag{3.14}$$

ND21 and ND22 are the circuits shown in Figure 3.11. Figure 3.11(a) has invalid sub-circuits. It can be concluded that

$$\begin{aligned} \mathtt{ND} &= (-1)^{g_o+j_o+c_u+d_u} \times \mathtt{ND11} + (-1)^{g_o+j_o+b_u+f_u+1} \times \mathtt{ND22} \\ &= (-1)^{c_u+d_u+1} \times \mathtt{ND11} + (-1)^{b_u+f_u} \times \mathtt{ND22}, \end{aligned} \tag{3.15}$$

for $g_o + j_o$ is an odd number. In this example, a 20-case decomposition approach is reduced to only 2 cases. Each case has well decomposed sub-circuits.

According to the above discussion and Theorem 1, if the sign adjustment of all

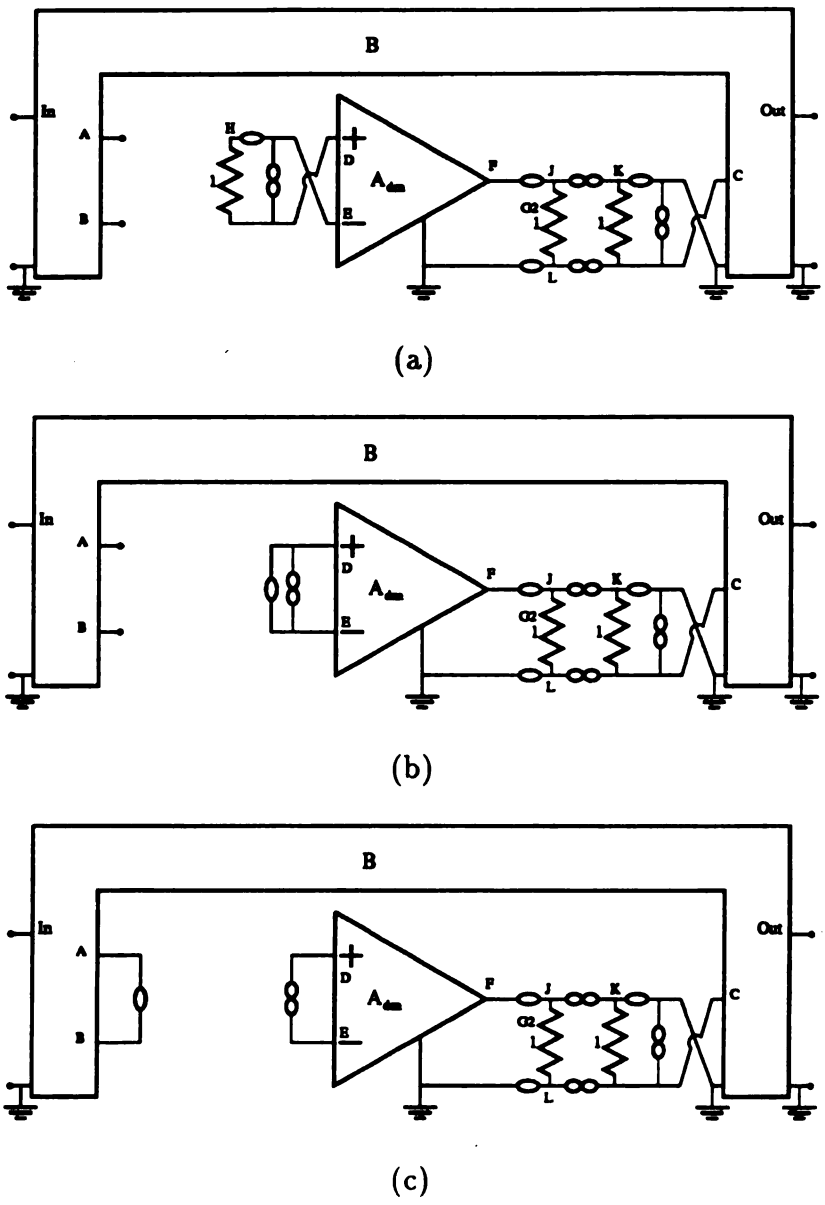
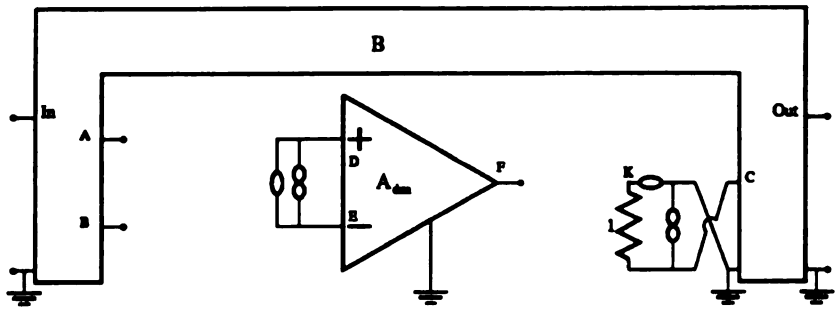
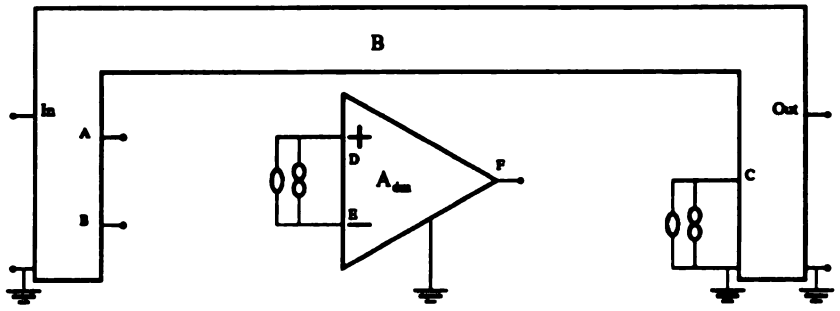


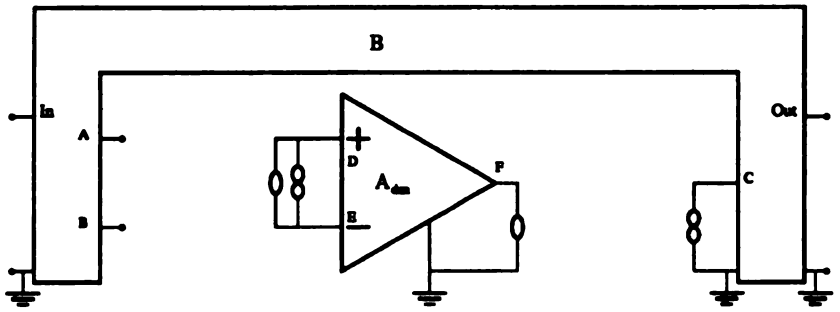
Figure 3.9. Cases expanded with respect to `G1`. (a) `ND1a`, Network Determinant without `G1`. (b) `ND1`, Further Simplification of (a). (c) `ND2`, Network Determinant with `G1`.



(a)

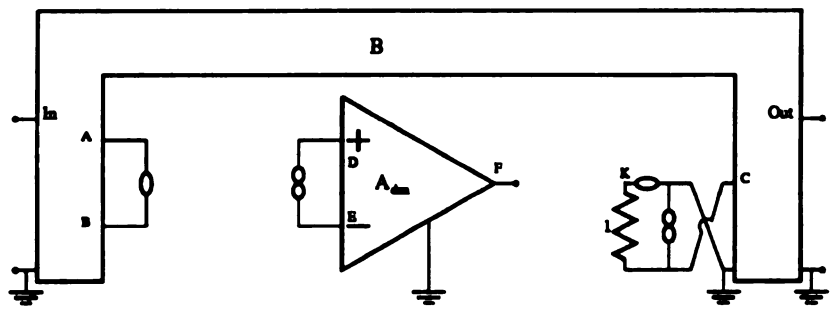


(b)

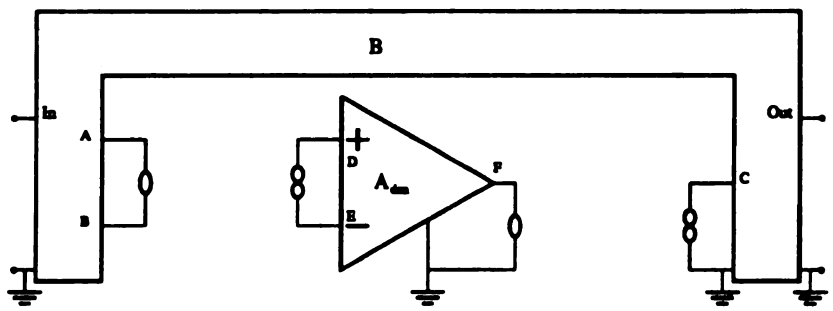


(c)

Figure 3.10. Cases expanded with respect to G2 of Figure 16(b). (a) ND11a, Network Determinant without G2. (b) ND11, Further Simplification of (a). (c) ND12, Network Determinant with G2.



(a)



(b)

Figure 3.11. Cases expanded with respect to G2 of Figure 16(c). (a) ND21, Network Determinant without G2. (b) ND22, Network Determinant with G2.

the network determinants of the decomposed cases is one, the transfer function of the circuit in Figure 3.7 is illustrated in Figure 3.12.

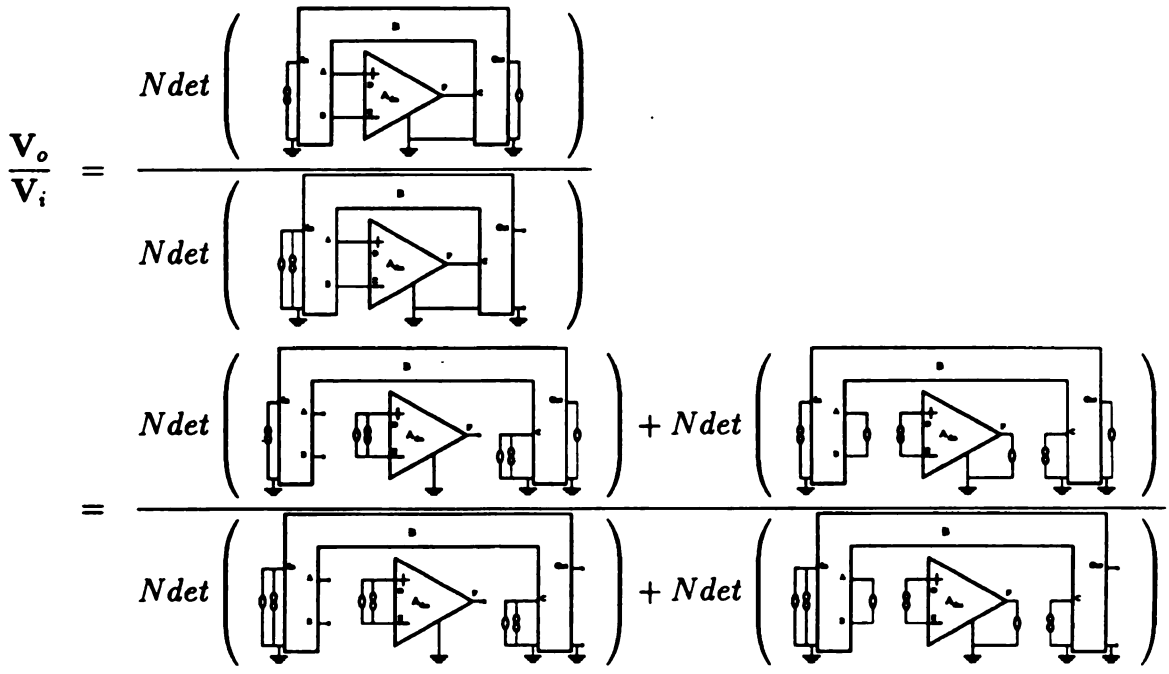


Figure 3.12. Network Determinant Equation of obtaining transfer function

Figure 3.13 shows a variation of the transfer function of Figure 3.12. It is done by applying a few simple algebraic operations so that its physical meanings can be understood from the transfer function. According to section 2.2, the transfer function of a circuit is the ratio of two network determinants which connect an extra nullator-norator pair to the input-output nodes and the ground of the circuit. The K, α, and β in Figure 3.13 are ratios of network determinants. Therefore, they are the transfer functions of the specific circuits.

According to Figure 3.13, K is the transfer function of the circuit in Figure 3.7 with its $\mathbf{A}_{dm}$ replaced by an ideal op-amp because an ideal op-amp can be modeled by

$$\frac{\mathbf{V}_o}{\mathbf{V}_i} = \frac{Ndet\left(\quad\right)}{Ndet\left(\quad\right)} \times$$

$$\frac{1 + \dfrac{Ndet\left(\quad\right)}{Ndet\left(\quad\right)} \times \dfrac{Ndet\left(\quad\right)}{Ndet\left(\quad\right)}}{1 + \dfrac{Ndet\left(\quad\right)}{Ndet\left(\quad\right)} \times \dfrac{Ndet\left(\quad\right)}{Ndet\left(\quad\right)}}$$

$$= \mathbf{K} \times \frac{1 + \frac{1}{\mathbf{A}_{dm}} \times \frac{1}{\alpha}}{1 + \frac{1}{\mathbf{A}_{dm}} \times \frac{1}{\beta}}$$

Figure 3.13. α, β network representation of the transfer function.

a nullator-norator pair. This is shown in Figure 3.14. α of Figure 3.13 is the transfer function of B in Figure 3.7, accomplished by changing node C to be the new input, node A and B to be the new output, and by connecting an ideal op-amp with its input at the old Output of B and with its output at the old Input of B. This is shown in Figure 3.15. Similarly, β is the transfer function of B accomplished by changing node C to be the input, node A and B to be the output, and short the original Input node to the ground. This is shown in Figure 3.16. Each of these figures represents a unique circuit. Also, according to Theorem 5, the $\mathbf{A}_{dm}$ represents the transfer function of the voltage controlled voltage source in Figure 3.7.

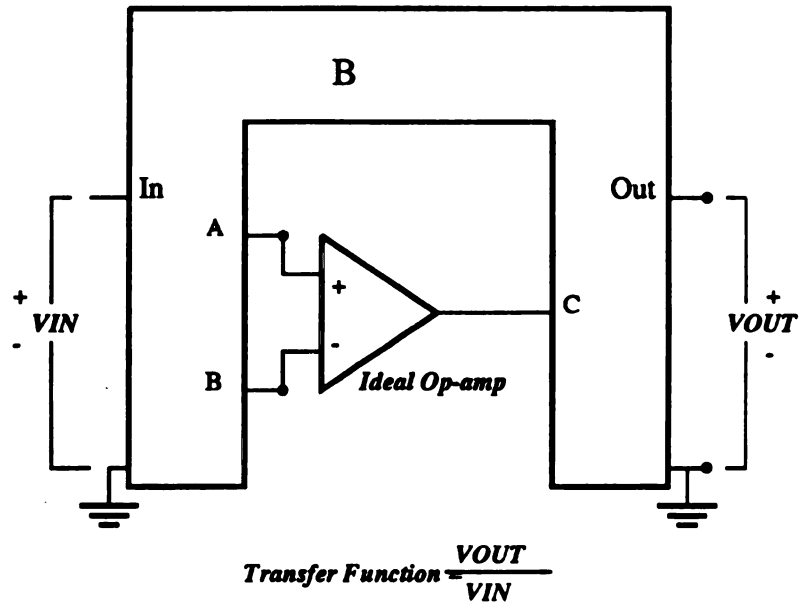


Figure 3.14. Schematic diagram of the transfer function **K**.

Through the above interpretation of Figure 3.13, one can find the sources of non-ideal effects of a circuit. Then, a circuit designer can concentrate on the decomposed α and β networks.

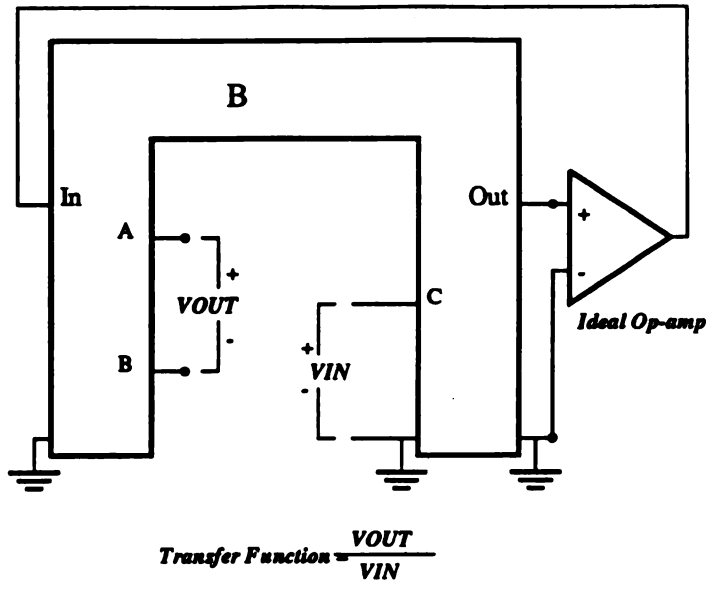


Figure 3.15. Schematic diagram of the transfer function α.

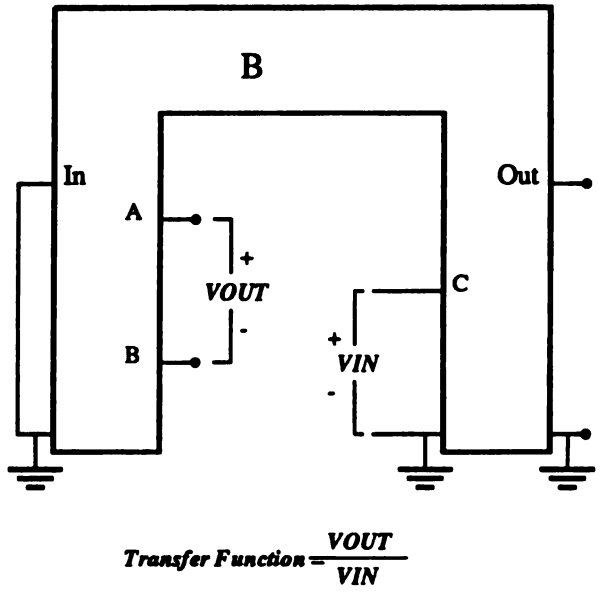


Figure 3.16. Schematic diagram of the transfer function β.

CHAPTER 4

NUMERICAL APPROXIMATION STRATEGIES FOR SYMBOLIC CIRCUIT ANALYSIS

The use of symbolic expressions to characterize input-output relations is an important analytic tool with a wide range of applications in the analysis and synthesis of networks and systems. However, the use of computer programs to accomplish symbolic analysis has some inherent problems related to memory consumption as well as to the computation inefficiency of obtaining network determinants addressed in chapter 2 and 3. The need for huge memory space for a moderate size commercial chip would eventually further worsen the computation efficiency. An greater problem is the large amount of output generated for the results of the circuits with more than 20 components. The volume of output generated in the symbolic analysis process currently represents one of the most restrictive limitations on the symbolic analysis technique. In this chapter, the proposed solutions are presented for different stages of the symbolic analysis process, which include numerical analysis before, during, and

after computation.

4.1 Numerical Approximation After Computation

The need for numerical approximation after computation is illustrated in the following example. Mathematically, it is trading off accuracy for simplicity.

Example 9 *Use symbolic approach to find the transfer function of the CMOS op-amp in Example 2.*

Answer: For simplicity, the low frequency small signal CMOS model is used in this example. The component values of the small signal model of each transistor is obtained from Pspice by using `.OP` card. A symbolic program [4] would generate the following results, where V6 represents the open loop gain.

```
Numerator of: V6

TERMS SORTED ACCORDING TO POWERS OF s

s**1 terms:

 + sC1*GM8*GM4*GM2*GM1 + sC1*GM8*GM4*GM2*GDS1
 + sC1*GM8*GM3*GM2*GM1 + sC1*GM8*GM3*GM2*GDS5
                .
                .
                .
      16 lines not shown

************************************************

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT
```

```
+ 2.54343e-30 * s**1 - 1.94226e-22 * s**0

************************************************

Denominator of: V6

TERMS SORTED ACCORDING TO POWERS OF s

s**2 terms:

 - sCL*sC1*GM8*GM3*GM2 - sCL*sC1*GM8*GM3*GM1
 - sCL*sC1*GM8*GM3*GDS5 - sCL*sC1*GM8*GM3*GDS2
 - sCL*sC1*GM8*GM3*GDS1 - sCL*sC1*GM8*GM2*GDS3
 - sCL*sC1*GM8*GM2*GDS1 - sCL*sC1*GM8*GM1*GDS3
 - sCL*sC1*GM8*GDS5*GDS3 - sCL*sC1*GM8*GDS5*GDS1
 - sCL*sC1*GM8*GDS3*GDS2 - sCL*sC1*GM8*GDS3*GDS1
 - sCL*sC1*GM8*GDS2*GDS1 - sCL*sC1*GM3*GM2*GDS8
              .
              .
              .
      133 lines not shown

************************************************

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

- 1.79078e-36 * s**2 - 3.05371e-29 * s**1 - 1.14017e-26 * s**0

************************************************
```

Clearly, the full symbolic result is too complicated to be used. Since the values of the components in this op-amp are known, and the coefficient of each order of **s** is dominated by only a few terms, the above symbolic result can be approximated and simplified. The following is the result of using 5% approximation.

```
Numerator of: v6
```

```
TERMS SORTED ACCORDING TO POWERS OF s

s**1 terms:

 + sC1*GM4*GM2*GM1 + sC1*GM3*GM2*GM1

s**0 terms:

 - GM6*GM4*GM2*GM1 - GM6*GM3*GM2*GM1

**************************************************

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

+ 6.78776e-26 * s**1 - 5.18338e-18 * s**0

**************************************************

Denominator of: v6

TERMS SORTED ACCORDING TO POWERS OF s

s**2 terms:

 - sCL*sC1*GM3*GM2 - sCL*sC1*GM3*GM1

s**1 terms:

 - sC1*GM6*GM3*GM2 - sC1*GM6*GM3*GM1

s**0 terms:

 - GM4*GM1*GDS7*GDS2 - GM4*GM1*GDS6*GDS2 - GM3*GM2*GDS7*GDS4
 - GM3*GM2*GDS6*GDS4 - GM3*GM1*GDS7*GDS4 - GM3*GM1*GDS7*GDS2
 - GM3*GM1*GDS6*GDS4 - GM3*GM1*GDS6*GDS2

**************************************************
```

```
NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

- 4.74668e-32 * s**2 - 7.97443e-25 * s**1 - 3.029e-22 * s**0

************************************************
```

Now, the simplified result is ready for analyzing this op-amp. □

The procedure for performing numerical approximation involves tedious floating point operations and sorting. When the size of the symbolic result increases, the computation time needed increases. Therefore, many approximation strategies have been proposed.

4.1.1 Numerical Approximation in ISAAC

The numerical approximation algorithm implemented in ISAAC [5] [23] is briefly introduced in this section. ISAAC's approach inspired the development of the approximation method of Sspice version 2.0.

The error definition being used in ISAAC is the accumulated absolute error ϵ_A :

$$\epsilon_A = \left. \frac{\sum |t_i(\underline{x})|}{|g(\underline{x})|} \right|_{\underline{x}=\underline{x}_o}, \tag{4.1}$$

where $g(\underline{x})$ is the original expression, $t_i(\underline{x})$ are the pruned terms, and $\underline{x}_o$ is the point of evaluation. Notice that in the numerator, the absolute values are summed. The error ϵ_A used in ISAAC is an estimation of the effective error, because the numerical values of the terms may cancel each other at the numerator of equation (4.1).

The approximation algorithm of ISAAC consists of two steps [23]. First, all terms smaller than the fraction ϵ_{max} of the original expression's mean value are discarded;

i.e., all $t_i(\underline{x})$ terms are discarded for which

$$\frac{\sum |t_i(\underline{x}_o)|}{|g(\underline{x})|} \leq \frac{\epsilon_{max}}{n} \tag{4.2}$$

with n being the number of terms in the original expression $g(\underline{x})$ and ϵ_{max} the maximum error as supplied by the user. Then, the remaining terms are sorted according to their magnitude and the smallest terms are removed as long as $\epsilon < \epsilon_{max}$. This approximation can be done for a specific frequency or over the whole frequency range.

The numerical approximation approach implemented in ISAAC spends expensive resources on sorting and indicating errors specified by the circuit designer. But, the error estimation definition, which is based on the accumulation of the absolute values of the terms that are given away, falls short of accomplishing its goal.

4.1.2 Numerical Approximation in SCOPE

SCOPE [24] is a Symbolic Circuit Output Processor and Evaluator. It allows the user to input numerical values for all the symbolic variables in order to generate a system function consisting of numerator and denominator polynomials in powers of s with numerical coefficients. This capability is also available in Sspice [4].

SCOPE also permits the user to retain any one of the symbolic variables as a symbolic quantity while numerical values are assigned to the other symbolic variables. This generates a symbolic input-output function consisting of numerator and denominator polynomials in powers of s with coefficients which are functions of a single symbolic variable. This is a mixed numerical and symbolic representation. However, SCOPE is limited to either using only one symbolic variable format or using all symbolic variable format expressions.

4.1.3 Numerical Approximation in Sspice

The numerical approximation technique of Sspice [4] is based on the fact that

$$\begin{aligned} f(s) &= \frac{1}{a+b} \\ &\approx \frac{1}{a} \end{aligned}$$

when $a >> b$. Sspice finds the largest term of each coefficient of orders of s. Then, throws away all the t_i terms in which

$$\frac{|t_i|}{|t_{largest}|} < \epsilon_{threshold}.$$

$\epsilon_{threshold}$ should be given by the user. This method avoids the complicated sorting mechanism which is necessary in ISAAC. Sspice version 2.0 provides the actual numerical error generated by the above procedure. This gives the circuit designer a concrete basis with which to evaluate the quality of their approximations.

Also, Sspice version 2.0 allows any number of symbols be substituted by their numerical values. This makes Sspice version 2.0 a complete mixed symbolic and numeric circuit analyzer and, thus, a more flexible one than SCOPE.

4.2 Numerical Approximation During Computation

In order to make symbolic circuit analysis practical for large circuits, effective numerical approximation is essential. In the past, the objectives of numerical approximation were focused on providing more informative answers to the circuit designer. Huge symbolic solutions were reduced to a few dominant terms so that critical circuit characteristics could be identified [25] [4] [5]. This approach has been successfully

implemented in many symbolic analog circuit analyzers. It is called numerical approximation after computation.

Through the usage and implementation of Sspice, it has been found that full symbolic computation is highly memory consuming. Without a good approximation strategy during computation, most of the existing computing systems cannot handle a circuit like an op-amp or a power supply regulator, because of the limited physical memory available. If we could approximate the intermediate results during computation, then the symbolic approach of circuit analysis would become practical.

Two factors affect the applicability of a numerical approximation strategy : efficiency and reliability. Traditional numerical approximation techniques involve searching, sorting, and high precision floating point computation. All these are highly time consuming. Executing the numerical approximation only once after obtaining a specific matrix determinant is acceptable. However, performing it many times during the determinant computation would not be desired. Since the matrix determinant can be obtained by decomposition, as described in section 3.2, numerical approximation can be applied only to the result of each sub-circuit, so that cpu time can be saved.

Furthermore, the existing approximation techniques, which throw away those unimportant terms, produce errors. If we apply the same technique during computation, the accumulation of errors may result in an unacceptable answer. The new technique , now, replaces the unimportant terms by their numerical values so that the accuracy of the results can be preserved. Therefore, the new approximation technique in Sspice makes this tool a mixed symbolic and numerical circuit analyzer.

The application of the new technique still needs more attention. The mixed numerical and symbolical solution may have terms which should be cancelled with full symbolic computation, now, left as a part of the solution. This is due to the truncation error of the floating point operation. Therefore, the user or the analyzer should check whether there exists duplicated component names [2] and circuit loops [9] . All

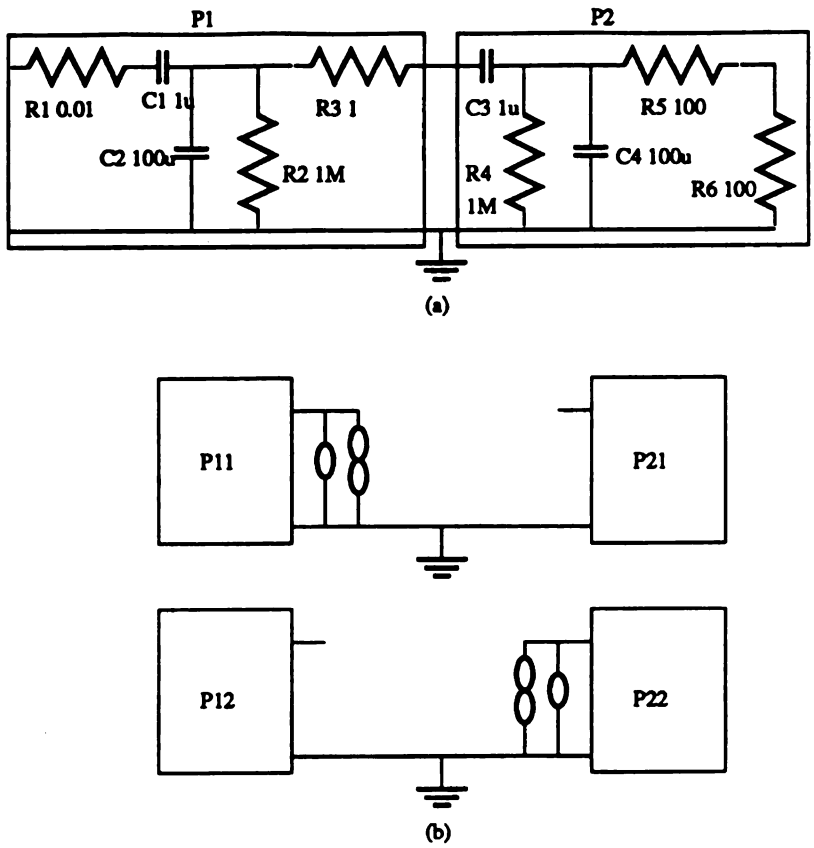


Figure 4.1. A Example for Approximation

these terms should be replaced by their numerical values as described for the new approximation strategy.

Example 10 *Figure 4.1 shows a circuit. Use the approximation during computation technique described above to find its network determinant.*

As shown in Figure 4.1(a), this circuit can be decomposed into P1 and P2. According to section 3.2, the network determinant is equal to $P11 \times P21 + P12 \times P22$, where P11, P12, P21, P22 are the network determinants of the decomposed sub-circuits shown in Figure 4.1(b). Table 4.1 shows the network determinants with and without approximation. The threshold value is 0.05. This example shows that the application of the new approximation method to the decomposed sub-circuits can maintain the

	without approx.	with approx.
P11	+sC2*sC1*G1 +sC1*G3*G1 +sC1*G2*G1	+sC2*sC1*G1 +sC1*G3*G1 +1e-10*s
P21	+sC4*sC3*G6 +sC4*sC3*G5 +sC3*G6*G5 +sC3*G6*G4 +sC3*G5*G4	+sC4*sC3*G6 +sC4*sC3*G5 +sC3*G6*G5 +2e-14*s
P12	+sC2*sC1*G3*G1 +sC1*G3*G2*G1	+sC2*sC1*G3*G1 +sC1*G3*G2*G1
P22	+sC4*G6 +sC4*G5 +sC3*G6 +sC3*G5 +G6*G5 +G6*G4 +G5*G4	+sC4*G6 +sC4*G5 +2e-8*s +G6*G5 +2e-8

Table 4.1. With and Without Approximation for P1 and P2

correct numerical values while reducing the number of terms for each case. □

4.3 Numerical Substitution Before Computation

For the approximation during computation strategy, the circuit designer can plug in the numerical values. They are identified or assumed not to be the critical component of a circuit before computing the matrix determinant. This is called numerical approximation before computation. This technique has been extremely successful for analyzing large circuits. Circuit designers can identify the critical components via their experiences and then verify their theory by running the circuit analyzer.

Example 15 in section 7.3 shows the application of this technique.

CHAPTER 5

SYMBOLIC SENSITIVITY ANALYSIS

In order to design a high-performance analog circuit, the designer should marshal every detail which may affect its functionality and performance. One of the most important measures which states the characteristics of a design is how sensitive the circuit is with respect to a specific element. Therefore, designers can design a more suitable circuit by trading off different factors. Sometimes, the changing of an element value may affect the sensitivity of other elements to a specific factor of merit. Therefore, a designer would appreciate not only to have the numerical value of the sensitivity measure but also its constitutions so that some trade offs can be considered more in-depth.

5.1 The Implementation of Symbolic Sensitivity Analysis

The calculation of sensitivity can be substantially simplified by applying the following rules which are called *Sensitivity Algebra* [3]. The symbolic sensitivity analysis, `SEN()` is done by applying the following rules recursively.

1. If $H = c$, where c is a constant, then $S_x^H = 0$.
2. If $H = cx$, then $S_x^H = 1$.
3. If $H = [cf(x)]^n$, then $S_x^H = nS_x^{f(x)}$.
4. If $H = f_1(x) + f_2(x) + ... + f_n(x)$, then $S_x^H = \frac{f_1 S_x^{f_1} + f_2 S_x^{f_2} + ... + f_n S_x^{f_n}}{f_1 + f_2 + ... + f_n}$.
5. If $H = f_1(x)f_2(x)...f_n(x)$, then $S_x^H = S_x^{f_1} + S_x^{f_2} + ... + S_x^{f_n}$.

The following is an example to show how these rules work.

Example 11 *If* $Q = \frac{1}{3-K}$, *then*

$$\begin{aligned} S_K^Q &= S_K^1 - S_K^{3-K} \\ &= -\frac{3S_K^3 + (-K)S_K^{-K}}{3-K} \\ &= \frac{K}{3-K} \end{aligned}$$

The symbolic network functions of a circuit are generally of the form

$$H(s) = \frac{N(s, p_1, p_2, ..., p_m)}{D(s, p_1, p_2, ..., p_m)}, \tag{5.1}$$

where N and D are both polynomials. Therefore, the sensitivity analysis formula becomes

$$\begin{aligned} S_p^H &= S_p^N - S_p^D = \frac{N_p}{N} - \frac{D_p}{D} \\ &= \frac{N_p \times D - D_p \times N}{N \times D}, \end{aligned} \tag{5.2}$$

where N_p is a polynomial that includes the element p in every terms. The polynomial of D_p also has the same property. Usually, the numbers of the terms in N and D are much greater than those in N_p and D_p, respectively. Therefore, the computation of $N \times D$ is very costly. If the user is choosing the numerical approximation option, then we can take advantage of approximating N and D before multiplying them to

get the denominator of Equation (5.2). Therefore, `QSN()` is implemented according to the following *Quick Sensitivity Algorithm.*

```
Quick Sensitivity Algorithm : QSN(N/D,p)
Np = NUM(SEN(N,p))
Dp = NUM(SEN(D,p))
If approximation is selected
Then {   Na = Significant terms in N;
         Da = Significant terms in D; }
Else {   Na = N;
         Da = D; }
Return( (Np*D-Dp*N)/(Na*Da) );
```

One may also notice that `QSN(H,p)` improve the speed of the computation without sacrificing the accuracy of other than the threshold value used for numerical approximation, set by the user. However, `SEN(SMY(H),p)` may suffer unexpected errors.

Finally, `DIF(H,x)` is implemented as `SEN(H,x)*H/x`.

5.2 Applications of Sensitivity Analysis

Figure 5.1 is a Tow-Thomas active filter.

We would like to evaluate the quality of this design.

By using Sspice, the voltage of node 2, say `V2`, can be obtained as shown in Equation 5.3 with $V_s = 1$.

$$V2 = \frac{sC2 * G6 * G4}{-sC2 * sC1 * G4 - sC2 * G4 * G1 - G5 * G3 * G2}. \tag{5.3}$$

Figure 5.2 shows its frequency response.

Sspice can identify that V2 is a bandpass filter function and provides its `Wo` and

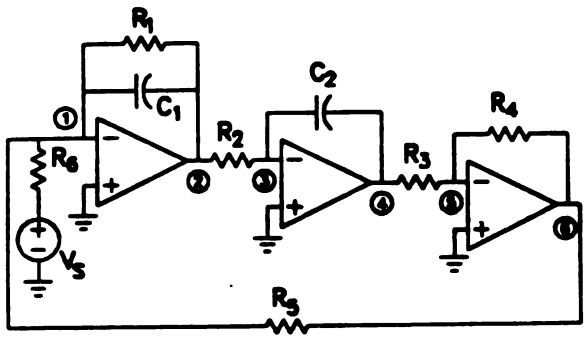


Figure 5.1. Tow-Thomas Active Filter

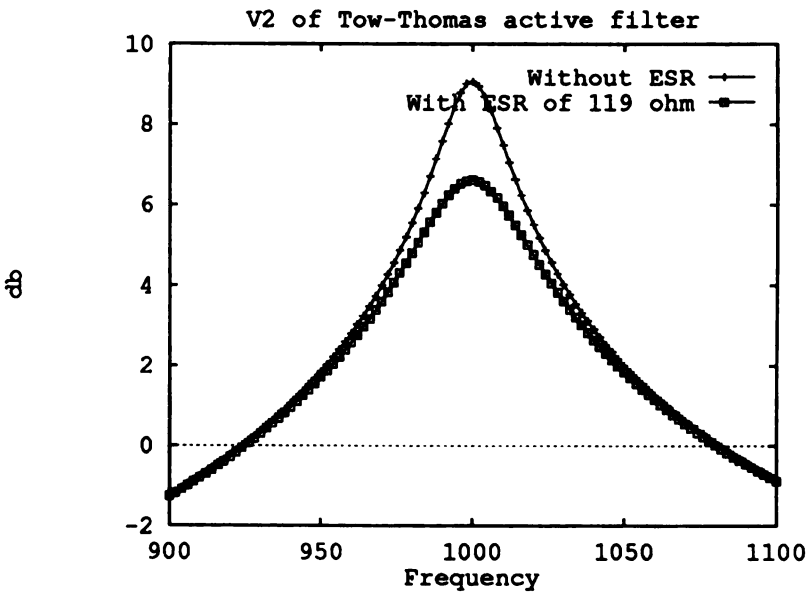


Figure 5.2. Bandpass filter function of V2.

Qo. They are

$$W_o = \sqrt{\frac{G5 * G3 * G2}{C2 * C1 * G4}}, and \tag{5.4}$$

$$Q_o = \frac{\sqrt{C1 * G5 * G3 * G2}}{G1 * \sqrt{C2 * G4}}. \tag{5.5}$$

Also, by the use of $\mathbf{S_{G3}^{V2}}$, one can find how resistor R3 affects the function of V2. Sspice can give the following equation,

$$S_{G3}^{V2} = \frac{G5 * G3 * G2}{-sC2 * sC1 * G4 - sC2 * G4 * G1 - G5 * G3 * G2} \tag{5.6}$$

which shows that R3 will affect V2 more at low frequencies than at very high frequencies. Figure 5.3 shows the plot.

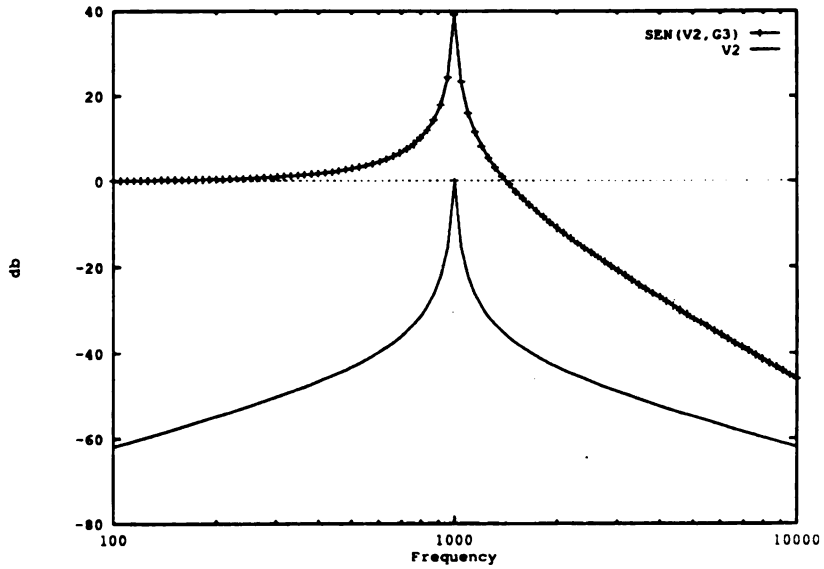


Figure 5.3. Sensitivity of V2 with respect to R3.

According to Equation 5.6, a designer can suppress S_{G3}^{V2} by increasing the value of G4. On the other hand, Equations 5.4 and 5.5 show that C1 and C2 can affect the W_o and Q_o of the filter function V2.

An interesting problem is how the imperfections of the capacitors affect the filter function. Ideally, the admittance of a capacitor has a phase angle of 90 degree. However, in reality, a capacitor may have a dissipation factor other than 0, which is usually modeled as a series resistor. We can utilize the sensitivity analysis capability of Sspice to study how dissipation factors affect the circuit. Suppose the capacitors of Figure 5.1 are mylar capacitors, the ESR would be around 119 Ω, which represents a dissipation factor of 0.0075. The following table shows the dissipation factors of different capacitors.

DISSIPATION FACTOR	
MYLAR	0.0075
CERAMIC(NPO)	0.0002
TANTALUM	0.04

Figure 5.2 shows the nonideal effect to the bandpass filter function of V2.

The following is an Sspice input file for the Tow-Thomas active filter. Rs1 and Rs2 are the series resistors of the nonideal capacitor C1 and C2, respectively.

```
Tow-Thomas Active Filter
VS 7 0 AC 1
R1 1 2 806K
R2 2 3 4K
R3 4 5 7.96K
R4 5 6 1K
R5 1 6 7.96K
R6 1 7 100K
C1 19 2 0.01U
C2 20 4 0.01U
XOA1 0 1 2 IDEAL OP-AMP
XOA2 0 3 4 IDEAL OP-AMP
XOA3 0 5 6 IDEAL OP-AMP
Rs2 3 20 1
Rs1 1 19 1
.END
```

Then, Sspice can give the expressions for W_o, Q_o and their sensitivities with respect to GS1 and GS2. The following are the Sspice printouts and their plots. With this printout, symbolic solutions can be obtained.

```
Numerator of: Wo**2=TRM(DEN(V2),0)/TRM(DEN(V2),2)

 + GS2*GS1*G5*G3*G2

Denominator of: Wo**2=TRM(DEN(V2),0)/TRM(DEN(V2),2)

 + GS2*GS1*G4*C2*C1 + GS2*G4*G1*C2*C1 + G5*G3*G2*C2*C1

****************************************

Numerator of: SEN(Wo**2,GS1)

 + GS2*G4*G1 + G5*G3*G2

Denominator of: SEN(Wo**2,GS1)

 + GS2*GS1*G4 + GS2*G4*G1 + G5*G3*G2

****************************************

Numerator of: Qo**2

 + GS2*GS2*GS1*GS1*G5*G4*G3*G2*C2*C1
 + GS2*GS2*GS1*G5*G4*G3*G2*G1*C2*C1
 + GS2*GS1*G5*G5*G3*G3*G2*G2*C2*C1

Denominator of: Qo**2

 + GS2*GS2*GS1*GS1*G4*G4*G1*G1*C2*C2
 + 2*GS2*GS2*GS1*G5*G4*G3*G2*G1*C2*C1
 + GS2*GS2*G5*G5*G3*G3*G2*G2*C1*C1
 + 2*GS2*GS1*GS1*G5*G4*G3*G2*G1*C2*C2
 + 2*GS2*GS1*G5*G5*G3*G3*G2*G2*C2*C1
```

```
 + GS1*GS1*G5*G5*G3*G3*G2*G2*C2*C2

****************************************

Numerator of: SEN(Qo**2,GS1)

 + 2*GS2*GS2*GS2*GS1*GS1*G5*G4*G4*G3*G2*G1*C2*C1
 - GS2*GS2*GS2*GS1*GS1*G4*G4*G4*G1*G1*G1*C2*C2
 + 2*GS2*GS2*GS2*GS1*G5*G5*G4*G3*G3*G2*G2*C1*C1
 .
 .
 (Total 9 terms )

Denominator of: SEN(Qo**2,GS1)

 + GS2*GS2*GS2*GS1*GS1*GS1*G4*G4*G4*G1*G1*C2*C2
 + 2*GS2*GS2*GS2*GS1*GS1*G5*G4*G4*G3*G2*G1*C2*C1
 + GS2*GS2*GS2*GS1*GS1*G4*G4*G4*G1*G1*G1*C2*C2
 .
 .
 (Total 15 terms )
```

The characteristics of Tow-Thomas filter with respect to the ESR, therefore, can be illustrated in Figure 5.4, Figure 5.5, Figure 5.6, and Figure 5.7. These circuit analysis capabilities are not available with any other numerical SPICE like programs.

Another active filter of the same class is the State Variable Active Filter, which is shown in Figure 5.8[4]. With V_{in}=1, Sspice can identify that V6 is a bandpass filter which is very similar to V2 of the Tow-Thomas active filter in Figure 5.1.

In the same way, we can study the Qo and Wo of the filter and their sensitivities

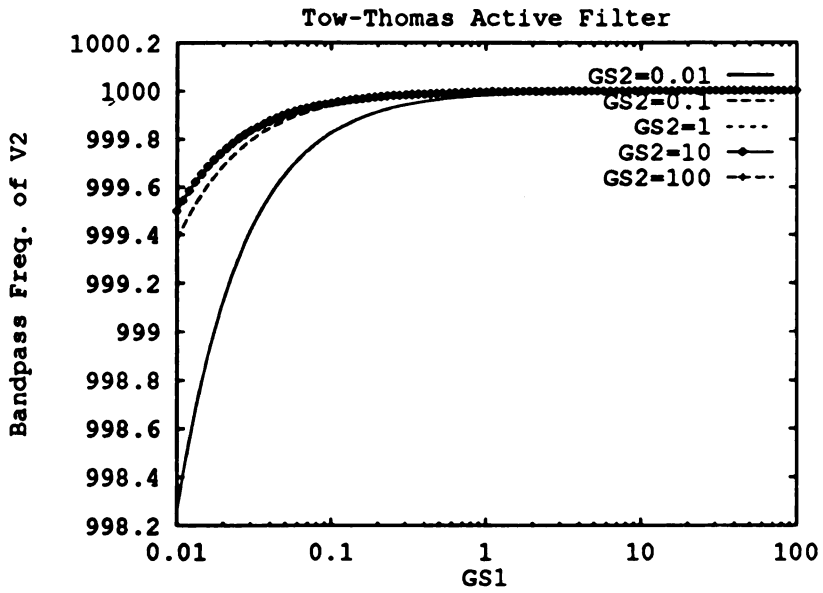


Figure 5.4. Wo of Tow-Thomas Filter.

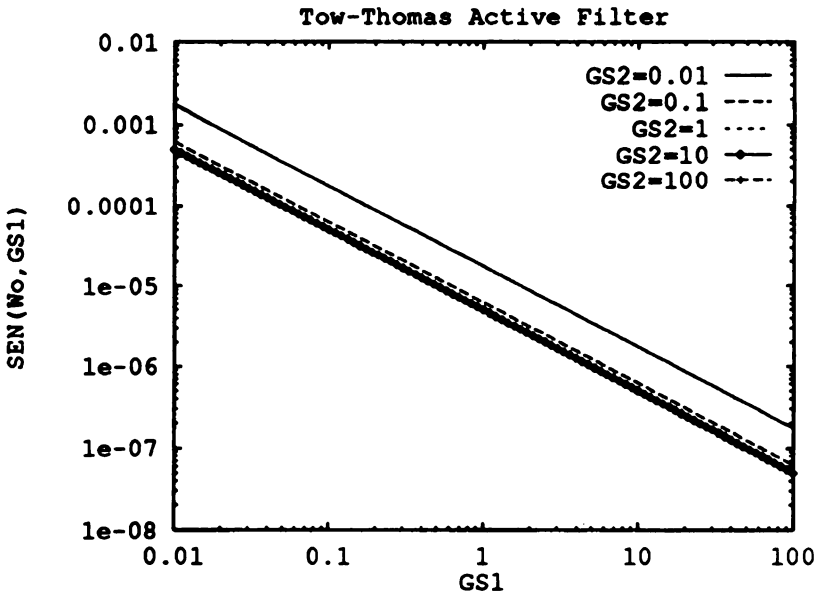


Figure 5.5. Sensitivity of Wo of Tow-Thomas Filter.

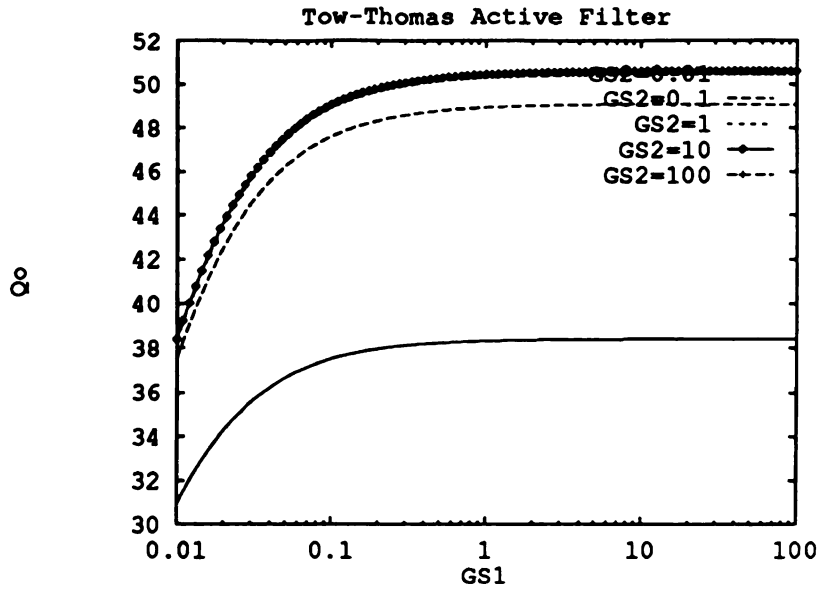


Figure 5.6. Qo of Tow-Thomas Filter.

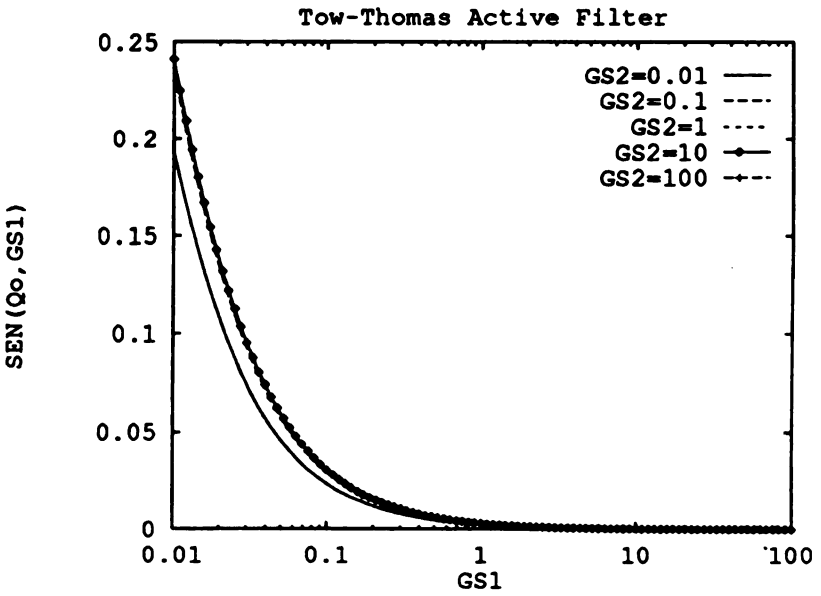


Figure 5.7. Sensitivity of Qo of Tow-Thomas Filter.

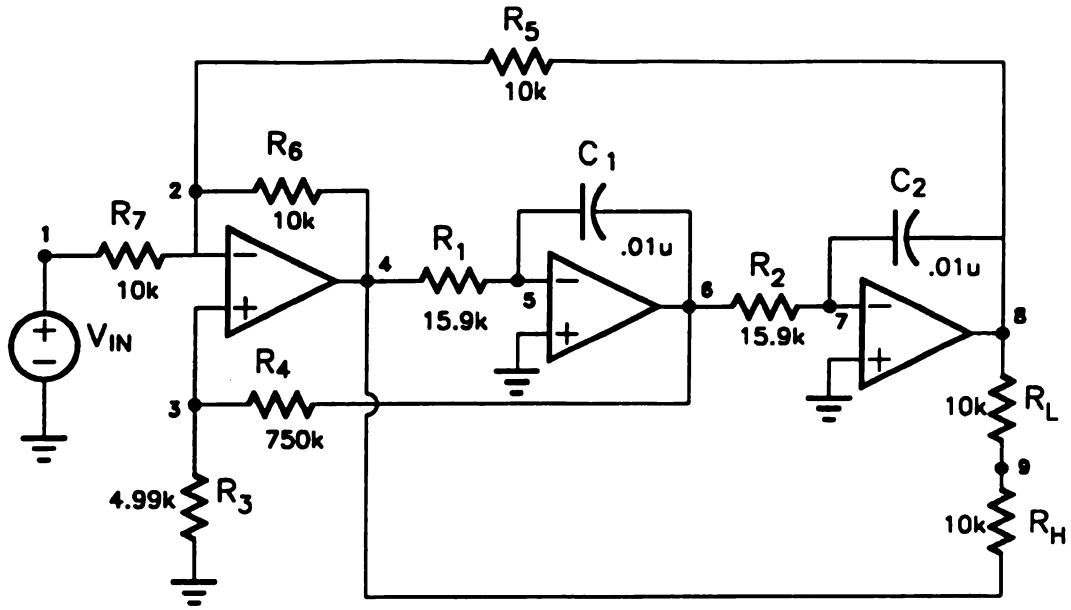


Figure 5.8. State Variable Active Filter

when there exist nonideal capacitors. Figure 5.10 demonstrates the value of Qo.

Computing the Qo sensitivity with respect to GS1, which is the series resistor modeling the dissipation factor of C1, is a tedious job. This will result in a complicated equation, as shown in the following.

```
Numerator of: SEN(Qo**2,GS1)

 - 432*GSC2*GSC2*GSC2*GSC1*GSC1*G4*G4*G4*G4*G1*C2*C2
 + 96*GSC2*GSC2*GSC2*GSC1*GSC1*G4*G4*G4*G4*G1*C2*C1
 - 432*GSC2*GSC2*GSC2*GSC1*GSC1*G4*G4*G4*G3*G1*C2*C2
 .
 .
 (Total 37 terms )

Denominator of: SEN(Qo**2,GS1)

 + 144*GSC2*GSC2*GSC2*GSC1*GSC1*GSC1*G4*G4*G4*G4*C2*C2
 + 288*GSC2*GSC2*GSC2*GSC1*GSC1*GSC1*G4*G4*G4*G3*C2*C2
```

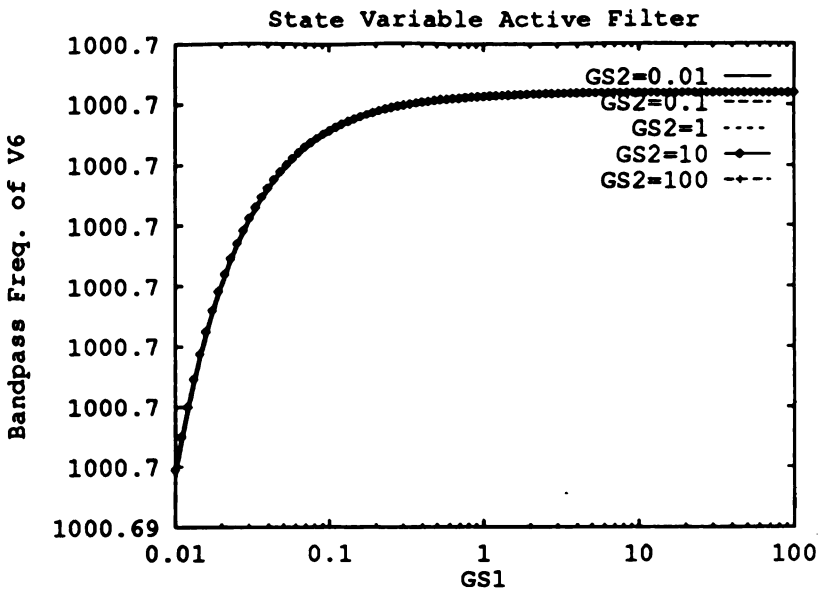


Figure 5.9. Wo of the State Variable Active Filter.

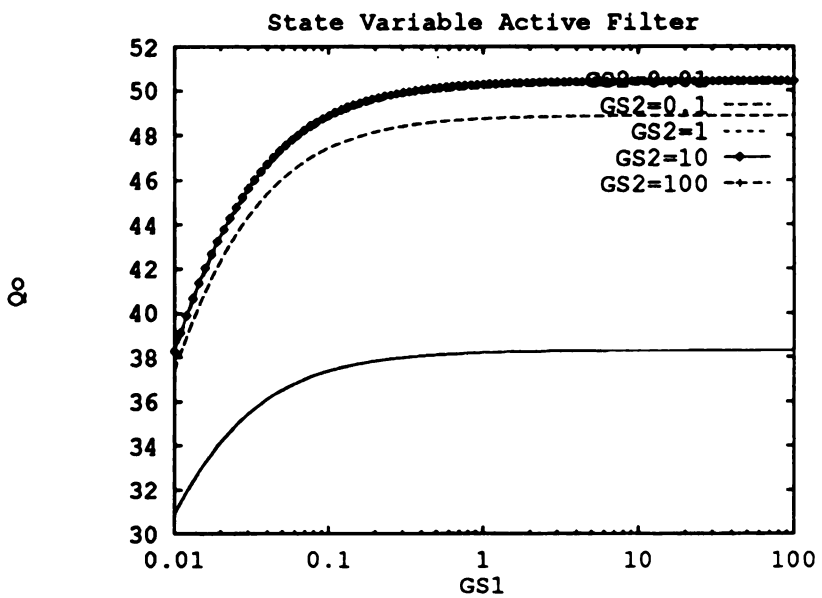


Figure 5.10. Qo of the State Variable Active Filter.

```
 + 144*GSC2*GSC2*GSC2*GSC1*GSC1*GSC1*G4*G4*G3*G3*C2*C2
 .
 .
 (Total 61 terms )
```

It is very hard to understand a complicated equation like the one above. With 1% approximation, Sspice can simplify the equation into a much simpler one, which is shown below.

```
Numerator of: QSN(Qo**2,GS1)

 + 288*G4*G3*G1*C1 + 96*G3*G3*G1*C1

Denominator of: QSN(Qo**2,GS1)

 + 288*GSC1*G4*G4*C2 + 144*GSC1*G4*G3*C2
```

Figure 5.11 shows that with 1% approximation, the sensitivity equation can be simplified with little sacrifice to the accuracy. We may find that the ESR of C2 is very insensitive to the sensitivity equation of Qo. Also, we can improve the Qo sensitivity with respect to the ESR of C1 by increasing C2 and decreasing C1.

With all the information provided above, we can conclude that, basically, the Tow-Thomas filter in Figure 5.1 is as good as the state variable active filter in Figure 5.8 in terms of its Qo sensitivity with respect to the nonideal capacitor.

Example 12 *Figure 5.8 shows the schematic of the State Variable active filter, where the transfer function at node 6 is a bandpass filter function. When* $R_1 = R_2$, *and* $R_5 = R_6 = R_7$ *always hold, this can be realized by a Programmable State Variable active filter chip set. The bandpass frequency is about 1000 Hz; and the Qo is about 50. Suppose C1 is a mylar capacitor which has the dissipation factor of 0.0075, this*

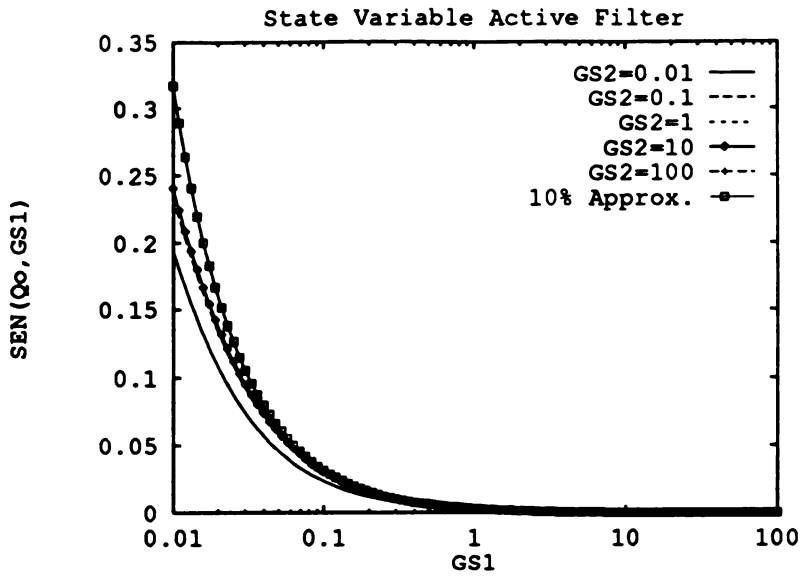


Figure 5.11. Sensitivity analysis without approximation

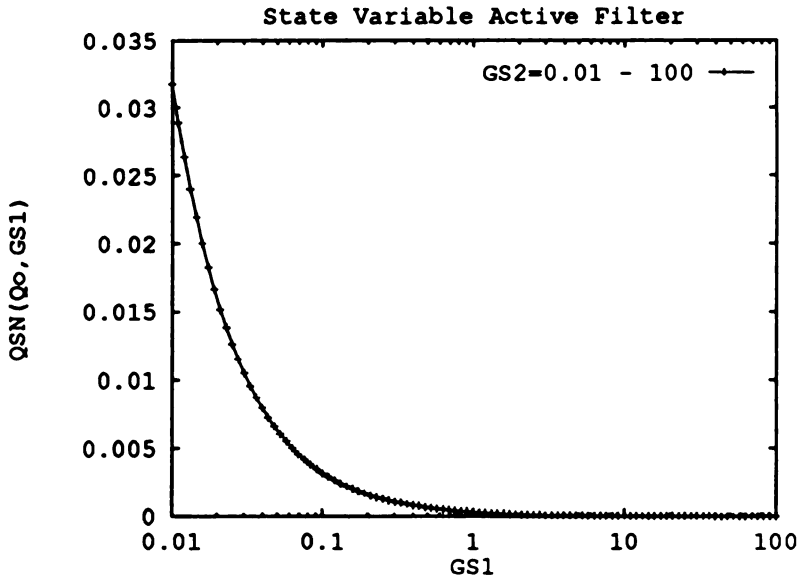


Figure 5.12. Sensitivity analysis with approximation

would produce deviations in Qo which are shown in Figure 5.13. This dissipation factor can be modeled by putting a series resistor of 119 Ω with C1. Is there any way that we can improve this Qo error by changing the values of other components without affecting the original design specifications while still using a mylar capacitor for C1?

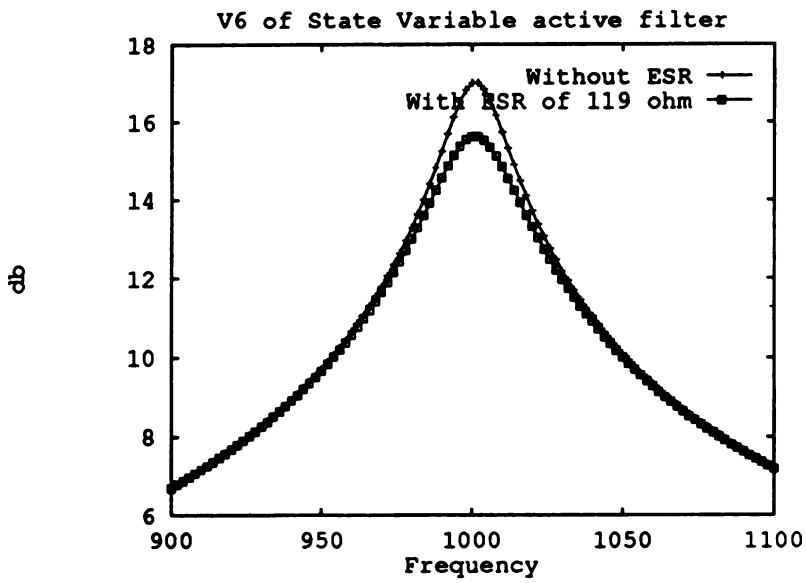


Figure 5.13. Bandpass filter function of V6 in Svaf

Answer: Usually, what a circuit designer can do is to use the design formulas for the ideal case to find an alternative design for the same specifications, then plug in the nonideal capacitor model to do the simulation. Every time the designer fails to pass the function verification, he really doesn't know how to make the next move, because it is too complicated.

By solving Qo sensitivity symbolically with 1% approximation, we find that

$$\mathbf{S}_{GS1}^{Q_o} = \frac{(288 * G4 + 96 * G3) * G3 * G1 * C1}{(288 * G4 + 144 * G3) * G4 * GS1 * C2 * 2}. \tag{5.7}$$

Because

$$\begin{aligned} G3 &>> G4, for \\ R3 &= 4.49K, and \\ R4 &= 750K, \end{aligned}$$

and

$$\begin{aligned} Q_o &= \frac{G3 \times \sqrt{C1}}{3 \times G4 \times \sqrt{C2}}, \\ W_o &= \frac{G1}{\sqrt{C1 \times C2}}, \\ D.F. &= \frac{GS1}{2\pi f_o \times C1}, \end{aligned}$$

then,

$$\mathbf{S}_{GS1}^{Q_o} = 9 \times D.F. \times Q_o, \tag{5.8}$$

where D.F. is the dissipation factor of C1. Therefore, there is no way that a circuit designer can accomplish the task stated in Example 12. □

CHAPTER 6

SYMBOLIC STABILITY ANALYSIS

The stability of an analog circuit is one of the most important considerations for the circuit designers. Usually, the verification by using a numerical circuit simulator doesn't provide enough information about the sources of instability. This leaves a great opportunity for the symbolic approach.

Since the symbolic circuit analyzer provides the transfer functions of a linear circuit in s domain, the stability of this circuit can be examined by doing the Hurwitz Test to the denominator of its transfer functions.

6.1 Hurwitz Test Fundamentals

It is well known that the denominator of the transfer functions of a stable system shouldn't have any zeros in the left half-plane and the zeros at the j-axis should be simple. The test for this property is known as a Hurwitz test, and the denominator is referred to as a Hurwitz polynomial. Before going into the details of this procedure, some of the important properties of Hurwitz polynomials are discussed.

Let the polynomial be written

$$Q(s) = a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n \tag{6.1}$$

with the even and odd parts

$$\begin{aligned} m(s) &= a_0 + a_2 s^2 + \cdots + a_n s^n \\ n(s) &= a_1 s + a_3 s^3 + \cdots + a_{n-1} s^{n-1} \end{aligned} \tag{6.2}$$

for n even, and with the last terms interchanged if the degree n is odd. $Q(s)$ is not a Hurwitz polynomial unless all coefficients are positive and no intermediate terms are missing. The only exception is the degenerate case in which the polynomial is an even or an odd function of s. Hurwitz polynomial can result only from the product of the following three kinds of factors

$$\begin{aligned} &(s + a) \text{ for } a \text{ real and positive,} \\ &(s^2 + b^2) \text{ for } b \text{ real,} \\ &(s^2 + 2as + a^2 + b^2) \text{ for } a \text{ and } b \text{ as above.} \end{aligned} \tag{6.3}$$

Let

$$\psi(s) = \frac{m(s)}{n(s)}. \tag{6.4}$$

The zeros of the polynomials, $m(s)$ and $n(s)$, are all simple and are restricted to line on the j-axis where they occur in conjugate pairs and alternate with each other. Thus $n(s)$, being odd, contains s as a factor and hence is zero at $s = 0$. The next largest pair of zeros in absolute value belong to the polynomial $m(s)$, the next largest are a pair of j-axis zeros contained in $n(s)$, and so forth. This situation is the alternation property of the zeros of m and n along the j-axis. Such a function, $\psi(s)$, is the

driving-point impedance or admittance of a lossless network - one containing only inductors and capacitors [26]. Suppose $\psi(s)$ is an impedance function and the order of the numerator is one degree higher than the denominator, then,

$$\psi'(s) = \psi(s) - \alpha_1 s \tag{6.5}$$

would still be a realizable LC driving-point impedance function. Similarly,

$$\psi''(s) = \frac{1}{\psi'(s)} - \alpha_2 s \tag{6.6}$$

would become a realizable LC driving-point admittance function.

The pattern of the test is thus established. If continued in the same manner, it leads to a continued-fraction expansion of the rational function $\psi(s)$ of the form

$$\psi(s) = \alpha_1 s + \cfrac{1}{\alpha_2 s + \cfrac{1}{\alpha_3 s + \cfrac{1}{\cdot + \cfrac{1}{\cdot + \cfrac{1}{\alpha_n s}}}}} \tag{6.7}$$

in which n is the degree of the original polynomial, and all coefficients $\alpha_1 \cdots \alpha_n$ must be positive if this polynomial is to have Hurwitz character.

The discussion so far as this testing procedure is concerned has assumed that the given polynomial, $Q(s)$, has no zeros on the j-axis. If it does, then the process of continued-fraction development will not continue for n terms but will terminate prematurely, and the j-axis factors will be placed in evidence at the point of termination. The reason for this behavior of the process is due to the fact that j-axis factors, as shown in set (6.4), have the form $(s^2 + b^2)$, which is an even function of s and hence if $Q(s) = m(s) + n(s)$ contains such a factor then it must separately be contained in both $m(s)$ and $n(s)$. In the rational function $\psi(s) = \frac{m(s)}{n(s)}$ such a common fac-

tor or factors cancels, and hence the continued division and inversion process would terminate sooner than it normally would. This process is similar to the procedure of obtaining the highest common factor of two given polynomials.

Because the derivative of a Hurwitz polynomial is again a Hurwitz polynomial, it is shown [26] that $(m+\frac{dm}{ds})$ and $(n+\frac{dn}{ds})$ are Hurwitz polynomials when $(m(s)+n(s))$ is a Hurwitz polynomial. This yield a simple procedure for determining whether a given even polynomial has only simple j-axis zeros, for if it has only such zeros then its derivative likewise has only such simple zeros which alternate with those of the given polynomial. An even Hurwitz polynomial divided by its derivative must then be a ψ function like Equation (6.7) and must yield a continued fraction with all positive coefficients. If this test fails, it may be concluded that the even polynomial does not have all simple j-axis zeros.

The above discussions complete the mathematical development of Hurwitz test.

6.2 Implementation of Hurwitz Test

The implementation of Hurwitz test in Sspice follows the procedure described in section 6.1. The polynomial under test is verified to know whether the ratio of its even part and its odd part is a realizable RC admittance or impedance function. If the continued-fraction procedure is terminated prematurely, the common factor of the even and odd polynomials should be an even polynomial to continue the test. Otherwise, the polynomial under test is not a Hurwitz polynomial. Then, the ratio of the above common factor and its first derivative should be a realizable RC function, which is tested by the same continued-fraction procedure only once. The following is the algorithm.

ALGORITHM : (Hurwitz Test)

Hurwitz_test(polynomial)

```
{
if( continued-fraction(polynomial,0)==Pass ) {
        return(Pass Hurwitz Test);
        }
else    {
return(Fail Hurwitz Test);
        }
}

continued-fraction(p,level)
/* p is the polynomial and level is an integer */
/* level==1 represents there exist j-axis zeros */
/* level==0 when this algorithm was called by   */
/* user's command                               */
{
p1=odd_polynomial(p);
p2=even_polynomial(p);
if(p1->degree<p2->degree) {
        p3=p1;
        p1=p2;
        p2=p3;
        }
p3=NULL;
while( p2 is not a zero polynomial ) {
        result=p1/p2;
        p3=result->remainder;
        p4=result->quotient;
        if( p4 is a polynomial other than s^1 only and
            the coefficient of s^1 is positive and
            the coefficient of s^0 is 0 ) {
                return(Fail);
                }
        p1=p2;
        p2=p3;
        }
if( p1 is not an even polynomial) {
        return(Fail);
```

```
        }
else    {
        if( p1 is larger than degree of 0 ) {
                if(level==1) {
                        /* Polynomial has multiple zeros */
                        /* at the j-axis                 */
                        return(Fail);
                        }
                p2=d(p1)/ds;
                p4=p1+p2;
                if(continued-fraction(p4,1)==Pass) {
                        return(Pass);
                        }
                else    {
                        return(Fail);
                        }
                }
        else    {
                return(Pass);
                }
        }
}
```

Example 13 *Figure 5.8 is a State Variable Active Filter. If the op-amps are all ideal, would this circuit always be stable when different component values are assigned when* $R_1 = R_2$, *and* $R_5 = R_6 = R_7$?

Answer: Applying the above algorithm, Sspice would give a report below.

```
Hurwitz Test of DEN(V8) :

The Following Rational Function Should Be >= 0

+2*G4*C1+2*G3*C1
R---------------R
+6*G4*G1
```

```
The Following Rational Function Should Be >= 0

+6*G4*C2
R--------R
+2*G4*G1+2*G3*G1


************************************************
The Following Polynominal PASS the Hurwitz Test
************************************************

TERMS SORTED ACCORDING TO POWERS OF s

s**2 terms: [0.000% error]

+2 * sC2*sC1*G3                                (99.34%)
+2 * sC2*sC1*G4                               (100.00%)

s**1 terms: [0.000% error]

+6 * sC2*G4*G1                                (100.00%)

s**0 terms: [0.000% error]

+2 * G3*G1*G1                                  (99.34%)
+2 * G4*G1*G1                                 (100.00%)

Operation of HTZ(DEN(V8)) :

        ********>> Hurwitz Test Completed <<********
```

Therefore, the necessary conditions for a stable State Variable Active Filter are

$$\frac{+2*G4*C1+2*G3*C1}{+6*G4*G1} \geq 0,$$

$$\frac{+6*G4*C2}{+2*G4*G1+2*G3*G1} \geq 0.$$

This is alway true in Figure 5.8 when $R_1 = R_2$, and $R_5 = R_6 = R_7$. Therefore, this is a stable arrangement for a programmable State Variable Active Filter when ideal op-amps are used. □

A circuit designer, thus, understands the stability of a circuit more in-depth so that he/she can make the best decision.

CHAPTER 7

IMPLEMENTATION OF SSPICE VERSION 2.0

7.1 Matrix Reduction Method

Nodal analysis is one of the most popular method for obtaining network functions. A circuit analyzer can layout the nodal equations in the form of

$$\mathbf{I} = \mathbf{Y} \times \mathbf{V}.$$

Therefore, the node voltages can be found by using Cramer's rule which requires the values of two matrix determinants. One matrix is the admittance matrix **Y**; the other is **Y** with a column replaced by **I** for the corresponding node.

Because, symbolic division is not preferred for *expanded format*, the matrix determinant is obtained by applying the following formula recursively :

$$det(\mathbf{Y}) = \sum_{i=1}^{n-1} (-1)^{i+1} \times G_{i,1} \times det(\mathbf{Y}_{i,1}), \tag{7.1}$$

where $det(\mathbf{Y}_{i,1})$ is the minor obtained by removing the first column and the i'th row

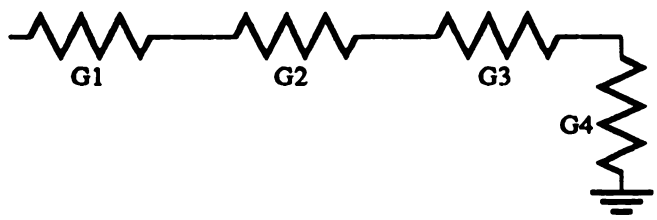


Figure 7.1. Series resistor circuit.

of $\mathbf{Y}$; and $G_{i,1}$ are the entries of the first column of $\mathbf{Y}$.

According to Equation (7.1), the computation complexity of obtaining matrix determinant depends not only on the dimension of the matrix but also the sparsity of the matrix. A matrix with less non-zero entries and less terms represents less computation and less cancellation while obtaining its determinant. This is because the additions and subtractions of a column or a row into another column or row within a matrix does not affect the determinant of the new matrix. If the application of these basic operations can make a matrix more sparse, then the computation complexity of find its determinant can be improved. The following is an example.

Example 14 *The admittance matrix of Figure 7.1 is*

$$\mathbf{Y} = \begin{bmatrix} G_1 & -G_1 & 0 & 0 \\ -G_1 & G_1 + G2 & -G_2 & 0 \\ 0 & -G_2 & G_2 + G_3 & -G_3 \\ 0 & 0 & -G_3 & G_3 + G_4 \end{bmatrix}. \tag{7.2}$$

Find the determinant of $\mathbf{Y}$.

Adding the first column of Equation (7.2) into the second column results in

$$det(\mathbf{Y}) = \begin{bmatrix} G_1 & 0 & 0 & 0 \\ -G_1 & G_2 & -G_2 & 0 \\ 0 & -G_2 & G_2+G_3 & -G_3 \\ 0 & 0 & -G_3 & G_3+G_4 \end{bmatrix}. \tag{7.3}$$

Similarly, adding the first row of Equation (7.3) into its second row would produce

$$det(\mathbf{Y}) = \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & -G_2 & 0 \\ 0 & -G_2 & G_2+G_3 & -G_3 \\ 0 & 0 & -G_3 & G_3+G_4 \end{bmatrix}. \tag{7.4}$$

If we repetitively apply these rules to rest of the columns and rows, we may produce that

$$det(\mathbf{Y}) = \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & G_4 \end{bmatrix}. \tag{7.5}$$

The computing effort needed for Equation (7.5) is, thus, far easier than that of Equation (7.2) when we apply Equation (7.1) for obtaining the determinant of $\mathbf{Y}$. □

Moreover, sparsity and the dimension of a matrix are not the only factors which affect the symbolic computation efficiency. In many cases, even though the application of the basic row or column operations cannot increase the number of zero entries in a matrix, the number of total terms may be decreased. This represents less cancellation during the computation.

Also, if a column has only one non-zero entry, $g_{i,j}$, which is located at the i'th

row, then all the other entries in the i'th row can be set to zero. Similarly, if a row has only one non-zero entry, $g_{i,j}$, then all the other entries in the j'th column can be set to zero. This may further ease the searching process in Equation (7.1).

This method has been found very effective for circuits with a dimension around 13 to 16. For example, the equivalent circuit of a chip bonding pad which has 14 components and whose dimension is 16, needs 77 seconds to produce its network determinant without the above matrix reduction algorithm using Sspice on a SPARC 1 workstation. However, with the application of the reduction technique, Sspice takes only 7.7 seconds to obtain the result.

7.2 Obtaining Matrix Determinant

Besides a good decomposition strategy and a matrix reduction method, an efficient algorithm for obtaining the determinant of an admittance matrix is equally important. This is especially true when there exists a tightly connected sub-circuit which is hard to be further decomposed. The determinant algorithm implemented in Sspice trades off between run time efficiency and memory consumption, which is very similar to the SLE/M algorithm in [23].

The determinant of a matrix $\mathbf{Y}$, $det(\mathbf{Y})$, can be calculated by the following formula :

$$det(\mathbf{Y}) = \sum_{i=1}^{n-1} (-1)^{i+1} \times G_{i,1} \times det(\mathbf{Y}_{i,1}), \tag{7.6}$$

where $det(\mathbf{Y}_{i,1})$ is the minor obtained by removing the first column and the i'th row of $\mathbf{Y}$; and $G_{i,1}$ are the entries of the first column of $\mathbf{Y}$. The minors can be calculated recursively according to Equation (7.6). There are relationships between different minors; they may share the same sub-minors. Figure 7.2 shows the relations among the minors of the admittance matrix when calculating its determinant.

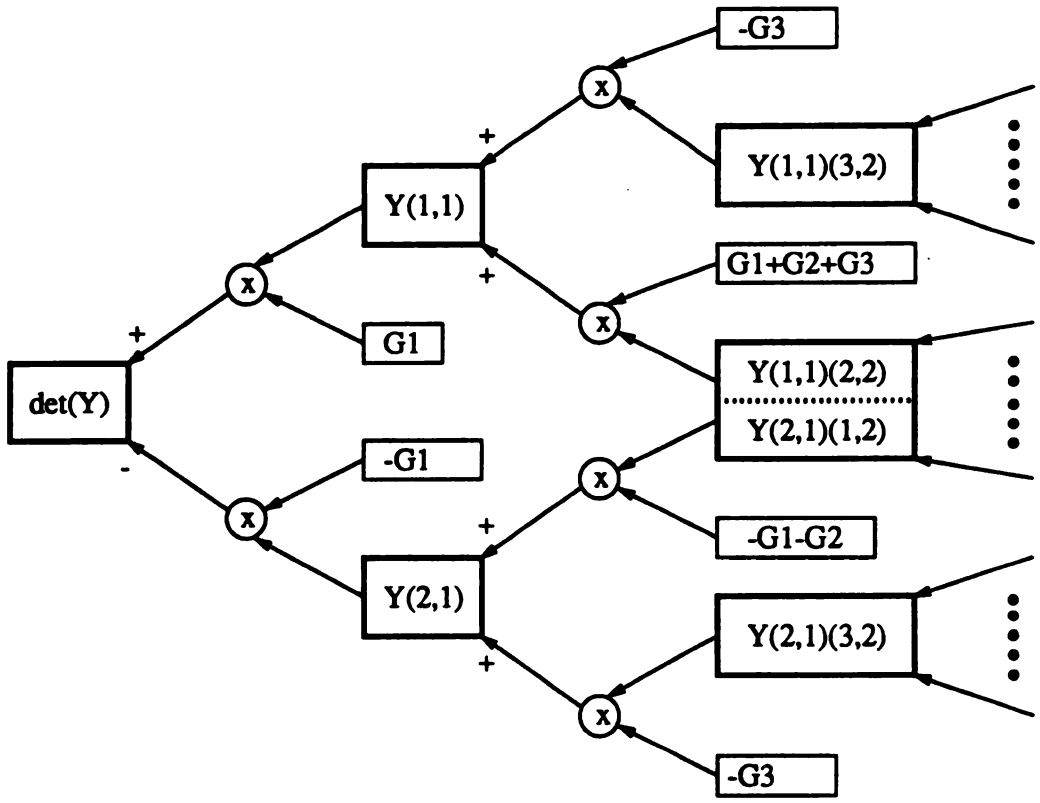


Figure 7.2. Relations between minors.

There are different levels of minors. We define $det(\mathbf{Y})$ itself to be the *0-level* minor. $\mathbf{Y}_{(i,j)}$ shown in Figure 7.2 represents a matrix with the i'th row and j'th column of $\mathbf{Y}$ eliminated and is called a *1-level* minor. Using a similar definition, $\mathbf{Y}_{(i,j),(p,q)}$ represents a matrix with i and p'th row and j and q'th column eliminated and is referred to as a *2-level* minor. For a matrix of dimension n, Sspice finds all the minors needed from *0-level* to *(n-1)-level*, and establishes the relations among different minors. It also identifies the many minors that are actually the same. For example, $\mathbf{Y}_{(i,j),(p,q)}$ and $\mathbf{Y}_{(p,j),(i,q)}$ are the same *2-level* minors. Finally, Sspice calculates the minors from the *(n-1)-level* down to *0-level.* The value of the *(n-1)-level* minors can be obtained from the entries of the n'th column of $\mathbf{Y}$ directly. Whenever Sspice completes the calculation of all the *i-level* minors, it releases the memory spaces occupied by the value of *(i+1)-level.* Because the symbolic values of the minors always consume a large amount of memory space, this strategy makes it possible to handle more symbols.

It is also found that both the numerator and the denominator of a transfer function may share the same sub-minors. Saving the values calculated for those minors may preserve computation efficiency. However, we find that, usually, the computation of the numerator needs much less time than that of the denominator. For example, the bandpass filter of the benchmark circuits in [27] takes 33 seconds with Sspice version 2.0 on a SPARC-1 station for the denominator and only 1.5 seconds for the numerator. Unlike the symbolic program described in [5], Sspice *does not* save the calculated minors with the exception of the *0-level* ones, in order to save memory space. Internally, Sspice calculates the denominator first, because the transfer functions at different nodes have the same denominator. It only needs to obtain the numerator when a specific transfer function is needed.

7.3 Implementation of Sspice

Sspice version 2.0 accepts the user's command interactively. Figure 7.3 shows the flow chart of Sspice. In this section, the advantage of using the decomposition approach is illustrated through the example of uA741 op-amp.

The default mode for Sspice version 2.0 is computation without decomposition. It checks the validity of the input files, aborting when the input circuit consists of loops of voltage sources, of nodes connected to branches which are all current sources, and so on. Then, it generates the nullator-norator equivalent circuit and the nodal equations for the full circuit. For transistors, a corresponding model as defined in the input SPICE file is substituted in automatically. When the decomposition procedure is issued by the user, the admittance matrices for the sub-circuits will be constructed.

One important mission for the circuit analyzer is to prepare critical information a circuit designer needs. Sspice has a well designed equation manipulator. It receives an equation from the user as a command, checks the syntax of this equation, puts the valid equation into a buffer and asks for the next equation. After the user gives all the functions, the equation manipulator begins to calculate the symbolic answers for all the equations together. Because the symbolic answers always take longer to obtain than the numerical ones, users may want to key in all the commands one at a time and let the computer run. If the user wants to get the current flow through the resistor `R1`, and `R1` is between node 3 and node 4, he/she can key in `"(V3-V4)*G1"`, where `G1=1/R1`, to get the answer. Also, he/she can type in `"SEN(V3-V4,G1)"` to do the sensitivity analysis of the voltage across `R1` with respect to itself. Users can also ask Sspice to plug in the numeric values of a set of components before the determinant computation is executed so that memory consumption and computation efficiency can be improved further. This produces partial symbolic solutions.

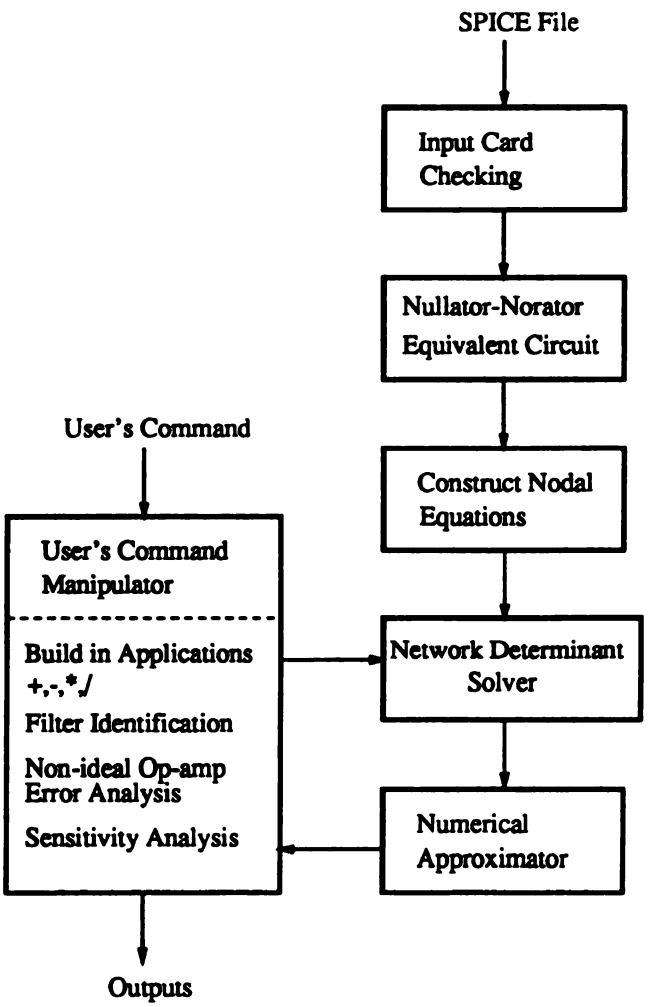


Figure 7.3. Flow Chart of Sspice.

For most situations of a medium size circuit, the symbolic solution can be overly long and circuit designers can find this hard to work with. Sspice can evaluate the numeric values for each term and throw away the insignificant terms. Usually, polynomials with hundreds of terms are dominated by only a few. Therefore, Sspice can do approximations for each coefficient of s according to a threshold value given by the user. The program can identify the most significant terms of an order of s, then eliminate those terms whose numerical values are smaller than the threshold value. By this method, the output of Sspice becomes more interpretable. Also, the errors due to this approximation are reported in the output.

Example 15 *Figure 7.4 shows the SPICE file of a 741 chip without compensation. The open loop gain is shown in Figure 7.5. This op-amp is unstable for closed loop gains less than 40db. How could we compensate this op-amp so that it becomes stable for all resistive feedback?*

Answer: By using Sspice, users can substitute in the high frequency transistor model [25] with a substrate capacitor. All the components are replaced by their numeric values except `Q16`, `Q17`, `Q23B`, and `R8`. The Laplace variable s is left as a symbol. Also, this circuit are decomposed at three nodes so that the computation efficiency is improved. The cut set are node number 9, 16, and ground. Sspice, thus, produces a circuit function of `s`, `G8`, and the high frequency model parameters of `Q23B`, `Q17`, and `Q16` as shown in Figure 7.6. According to the Sspice output in Figure 7.6, the location of the first pole can be formulated to be

$$\frac{1}{2\pi \times \texttt{CPI23B}}\left(\frac{1.5798 \times 10^{-5} \times \texttt{GPI17} \times \texttt{GPI16}}{\texttt{GM17} \times \texttt{GM16}} + \frac{1.1469 \times 10^{-12}}{\texttt{G8}} + \frac{2.1732 \times 10^{-10} \times \texttt{GPI16}}{\texttt{GM16} \times \texttt{G8}} + \frac{1.6858 \times 10^{-12}}{\texttt{GM17}} + \frac{3.1943 \times 10^{-10} \times \texttt{GPI16}}{\texttt{GM17} \times \texttt{GM16}}\right).$$

```
741A OPERATIONAL AMPLFIER SUBCIRCUIT
*.SUBCKT UA741A   4    5    1   13   25
*                Vi+ Vi- Vp+ Vp- Vout
Vin 100 0 AC 1
Voff 100 4 327U
Vinn 5 0 0
Vpp 1 0 DC 15
Vpn 13 0 DC -15
QH1 2 4 6 13 M1
QH2 2 5 7 13 M1
QH3 8 3 6 13 M2
QH4 9 3 7 13 M2
QH5 8 10 11 13 M1
QH6 9 10 12 13 M1
QH7 1 8 10 13 M1
QH8 2 2 1 13 M2
QH9 3 2 1 13 M2
QH10 3 15 14 13 M1
QH11 15 15 13 13 M1
QH12 16 16 1 13 M2
QH13A 21 16 1 13 M2 .25
QH13B 17 16 1 13 M2 .75
QH14 1 21 24 13 M1 3
QH15 21 24 25 13 M1
QH16 1 9 18 13 M1
QH17 17 18 19 13 M1
QH18 21 22 23 13 M1
QH19 21 21 22 13 M1
QH20 13 23 26 13 M2 3
QH21 20 26 25 13 M2
QH22 9 20 13 13 M1
QH23A 13 17 23 13 M2
QH23B 13 17 9 13 M2
QH24 20 20 13 13 M2
R1 11 13 1K
R2 12 13 1K
R3 10 13 50K
R4 14 13 5K
R5 16 15 39K
R6 24 25 27
R7 25 26 22
R8 19 13 100
R9 18 13 50K
R10 22 23 40K
R11 20 13 50K

.MODEL M1 NPN (BF=200 IS=1E-14 VAF=125  VJS=.75
+ RB=185 RC=15 CJE=.65P CJC=.36P TF=1.15N TR=405N CJS=3.2P MJS=.5)
.MODEL M2 LPNP (BF=50  IS=1E-14 VAF=50  VJS=.75
+ RB=500 RC=150 CJE=.1P CJC=1.05P TF=27.4N TR=2540N CJS=5.1P
+ MJS=.5)
.OP
.AC DEC 100 1 100MEG
.PROBE
.END
```

Figure 7.4. The SPICE file of 741 chip without compensation.

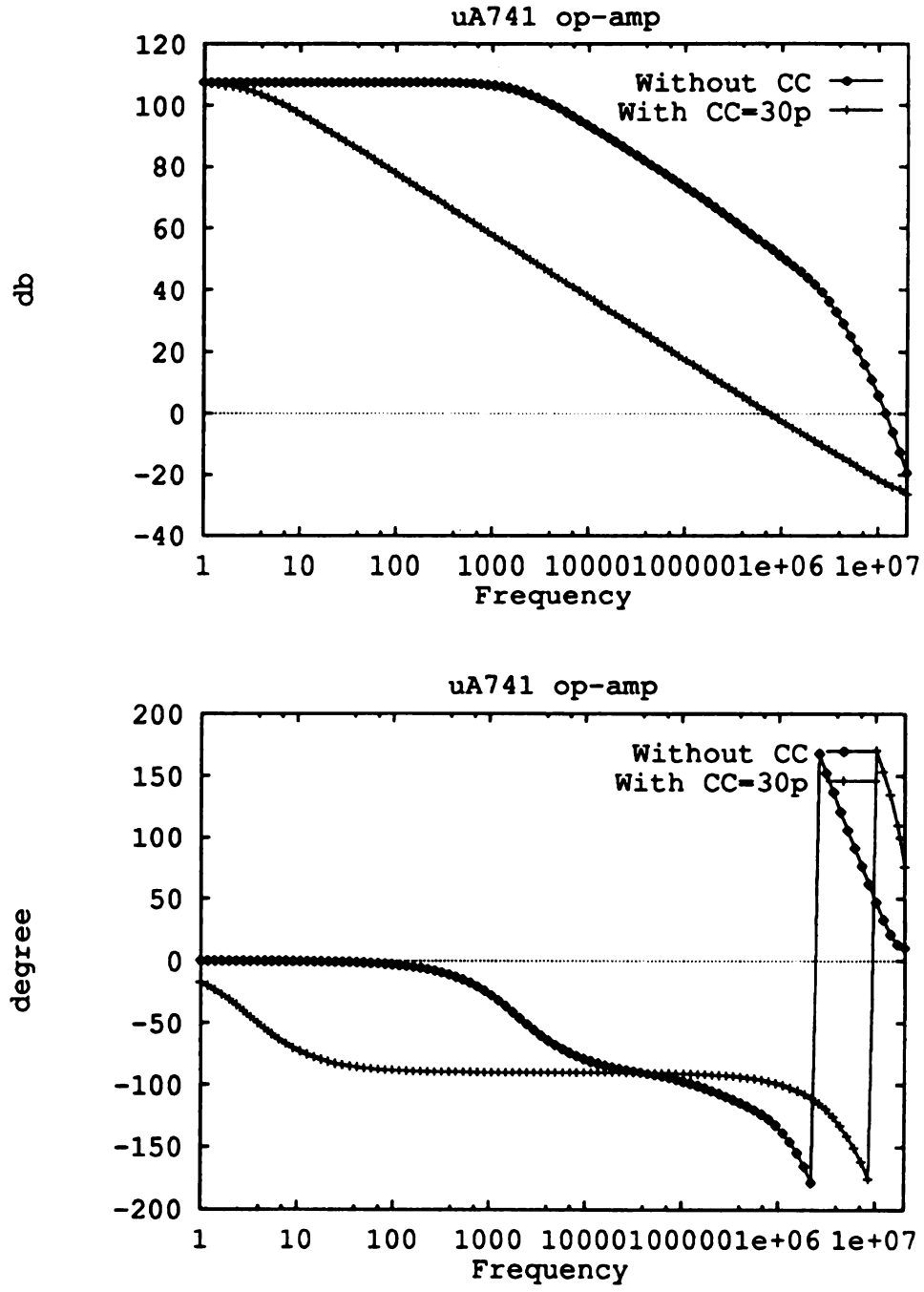


Figure 7.5. Frequency response of uA741 op-amp for gain and phase.

This op-amp can be made stable by decreasing the value of the first pole so that a reasonable phase margin is obtained at the frequency where the gain of the amplifier is unity. According to the above equation, the value of the first pole can be decreased by increasing the value of CPI23B. This is done by adding a capacitor between the emitter and base of Q23B. A 30pF capacitor, CC, between node 17 and 9, therefore, compensates this op-amp with a phase margin about 80 degree. The function of a compensated 741 chip is also shown in Figure 7.5. □

```
Numerator of: V25

+5.58152e-246 * s**22 +8.92052e-235 * s**21 +2.01726e-224 * s**20 +1.65705e-214 * s**19
+5.88645e-205 * s**18 +9.80216e-196 * s**17 +3.25670e-202 * s**16 +4.18728e-178 * s**15
+1.10986e-169 * s**14 +9.74666e-162 * s**13 -3.77414e-153 * s**12 -1.67351e-144 * s**11
-3.24763e-136 * s**10 -3.37697e-128 * s**9 -9.16272e-121 * s**8 +2.62565e-112 * s**7 +4.
28602e-104 * s**6 -7.02128e-101 * s**5 +1.50833e-88 * s**4 +4.37042e-81 * s**3 +7.97454e
-74 * s**2 +8.40643e-67 * s**1 +3.82170e-60 * s**0

****************************************

Denominator of: V25

TERMS SORTED ACCORDING TO POWERS OF s

s**22 terms: [0.000% error]

+0.474915e-233 * sCC*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s  (97.33%)
+0.390420e-245 * s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s*s  (100.00%)
.
.
s**1 terms: [0.000% error]

+2.64415e-56 * sCC                                                  (99.86%)
+1.10508e-69 * s                                                    (100.00%)

s**0 terms: [0.000% error]

+1.63648e-65 * 1                                                    (100.00%)

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

+1.46379e-244 * s**22 +2.15324e-233 * s**21 +5.35456e-223 * s**20 +5.08259e-213 * s**19
+2.11331e-203 * s**18 +4.14079e-194 * s**17 +4.51969e-185 * s**16 +3.11176e-176 * s**15
+1.47609e-167 * s**14 +5.10698e-159 * s**13 +1.33037e-150 * s**12 +2.66099e-142 * s**11
+4.11177e-134 * s**10 +4.89797e-126 * s**9 +4.45561e-118 * s**8 +3.02822e-110 * s**7 +1.
49523e-102 * s**6 +5.22735e-95 * s**5 +1.26101e-87 * s**4 +2.00856e-80 * s**3 +1.90072e-
73 * s**2 +7.94350e-67 * s**1 +1.63648e-65 * s**0
```

Figure 7.6. Sspice output of open loop transfer function of 741 without compensation.

Without decomposition, a SPARC-2 workstation need 1 hour to finish the job. However, with the application of decomposition at node 9, 16, and the ground for the above example, Sspice version 2.0 accomplished the task within 4 minutes. Usually, circuit designers use trial-and-error to find the answer they need. This may require several runs of a symbolic simulator. This example demonstrates the importance of the composition and decomposition strategy.

7.4 Second Order Filter Function Identification

One of the most important circuit applications is signal processing. Analog filters play an important role for both analog and digital signal processings. In many cases, the characteristics of the analog filters determine the performance of the whole system. Therefore, analog filter function analysis is the first step toward a successful design. The primary function of a filter is to pass or to stop a band of frequencies between the input and output. Figure 7.7 shows the behavior of the magnitude of the ideal filter functions versus frequency. In each of these responses, there are pass bands and stop bands as the name of the filter indicated. The filters of Figure 7.7 represent the most desired response. Because circuits respond to a frequency with finite slopes, it is impossible to synthesize a circuit which has the ideal behavior. Circuit designer have to tradeoff among cost, performance, and other factors with an appropriate approximation. Second order approximation is one of the most commonly used techniques. Also, second order filter functions are the basic building blocks of many higher order filters. It would be valuable to circuit designers to have a tool to do in-depth analysis of second filter functions.

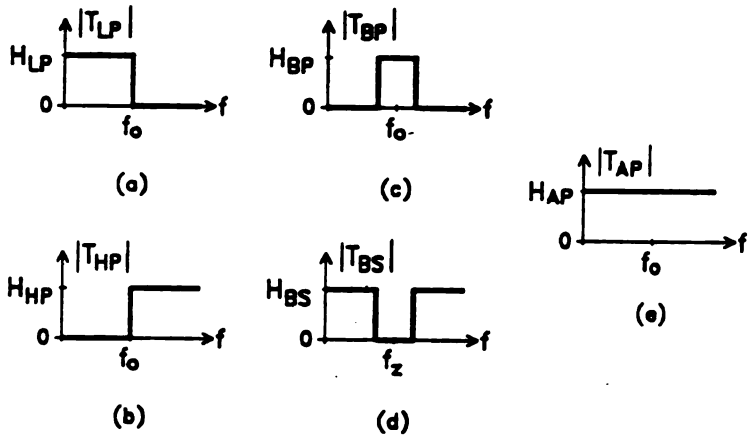


Figure 7.7. (a) Low Pass. (b) High Pass. (c) Band Pass. (d) Band Stop. (e) All Pass filter functions.

The second order transfer function of a low pass filter is

$$T_{LP}(s) = \frac{H_{lp}\omega_o^2}{s^2 + (\frac{\omega_o}{Q_o})s + \omega_o^2}. \tag{7.7}$$

The second order transfer function of a high pass filter is

$$T_{HP}(s) = \frac{H_{hp}s^2}{s^2 + (\frac{\omega_o}{Q_o})s + \omega_o^2}. \tag{7.8}$$

The second order transfer function of a band pass filter is

$$T_{BP}(s) = \frac{H_{bp}(\frac{\omega_o}{Q_o})s}{s^2 + (\frac{\omega_o}{Q_o})s + \omega_o^2}. \tag{7.9}$$

The second order transfer function of a band stop filter is

$$T_{BS}(s) = \frac{H_{bs}(s^2 + \omega_z^2)}{s^2 + (\frac{\omega_o}{Q_o})s + \omega_o^2}. \tag{7.10}$$

The second order transfer function of a all pass filter is

$$T_{AP}(s) = \frac{H_{ap}[s^2 - (\frac{\omega_o}{Q_o})s + \omega_o^2]}{s^2 + (\frac{\omega_o}{Q_o})s + \omega_o^2}. \tag{7.11}$$

Sspice has the capability of generating detailed symbolic equations for second filter functions like H_{ap} and ω_o. Also, these equations can be obtained with the special function command as described in section 7.6.

7.5 In-Band Error Approximation

Designing a filter circuit is usually done by specifying the filter parameters and then selecting a specific topological realization with resistor and capacitor values to meet these specifications. Circuit designers always consider the components are ideal so that the frame work of the design can be analyzed with much simplification. However, in building an active filter circuit with real components, there are deviations from the design parameters. One of the major contributors to errors is the nonideal effects of an op-amp. At very low frequencies, the open loop gain of an op-amp is approximated to be infinitely large. This is no longer true for frequencies larger than 50 kHz. Thus, the op-amp can be approximated by a single pole expression as follows :

$$A = \frac{A_o\omega_o}{s + \omega_o}, \tag{7.12}$$

where A_o is the dc gain and $A_o f_o$, or $\frac{A_o \omega_o}{2\pi}$, is the gain-bandwidth-product(GBP). For a TL084 quad op-amp, $A_o = 200k$ and $f_o = 22.2Hz$. By plugging in the one pole model, Sspice can calculate the Qo and Wo errors due to this dominant pole. The following is an example.

Example 16 *The following is a State Variable Active Filter using TL084 op-amps.*

```
State Variable Active Filter
VIN 1 0 AC 1
R5 1 2 10K
R5 2 4 10K
R3 3 0 4.99K
R4 3 6 750K
R5 2 8 10K
XNOA 3 2 4 TL084
R1 4 5 159K
C1 5 6 0.001U
XNOA 0 5 6 TL084
R1 6 7 159K
C2 7 8 0.001U
XNOA 0 7 8 TL084
RL 8 9 10K
RL 9 4 10K
.END
```

What is the Qo and fo errors of this filter in comparison with the same design using ideal op-amps.

Answer: Sspice would generate a report in the following.

```
State Variable Active Filter

             *************
             * Qo and fo *
             *************
```

Qo is:

```
SQRT{( + C1)*( + 4*G4*G4 + 8*G4*G3 + 4*G3*G3)}
-----------------------------------------------
( + 6*1)*( + G4)*SQRT{ + C2}
```

= 50.4337

(2*PI*fo)**2 is:

```
( + G1*G1)
------------
( + C2*C1)
```

fo = 1000.97 Hz

****
* DQo/Qo = (D2-D0)foQo - Dfo/fo *
****

D2-D0: where k{i} = 1/GBP{i}

The numerator is:

+k TIMES

+ 8*G4*G4 + 28*G4*G3 + 20*G3*G3

The denominator is:

+ 4*G4*G4 + 8*G4*G3 + 4*G3*G3

****
* Dfo/fo = (D2-D1)fo/(2Qo) *
****

D2-D1 : where k{i} = 1/GBP{i}

```
The numerator is:

+k TIMES

 + 44*C2*G4*G4 + 40*C2*G4*G3 - 4*C2*G3*G3 - 4*C1*G4*G4 - 8*C1*G4*G3
 - 4*C1*G3*G3

The denominator is:

( + G4*C2)*( + 12*G4 + 12*G3)

                ******************************
                *   NUMERICAL  EVALUATION    *
                ******************************

Dfo/fo = -0.000218473

DQo/Qo = 0.0573579
```

Therefore, both symbolic and numeric expressions of the Qo and fo errors are obtained. This is verified by using Pspice as shown in Figure 7.8. Both Sspice and Pspice outputs show that the errors due to the dominant pole are small.

□

7.6 Special Functions

Sspice has a built in equation editor to manipulate the answers for the users. For example, `V(1,2)` is interpreted as `V1-V2`. Right now, more functions have been implemented in Sspice, and many of them can be used as a part of the sensitivity analysis features. The followings are their brief descriptions.

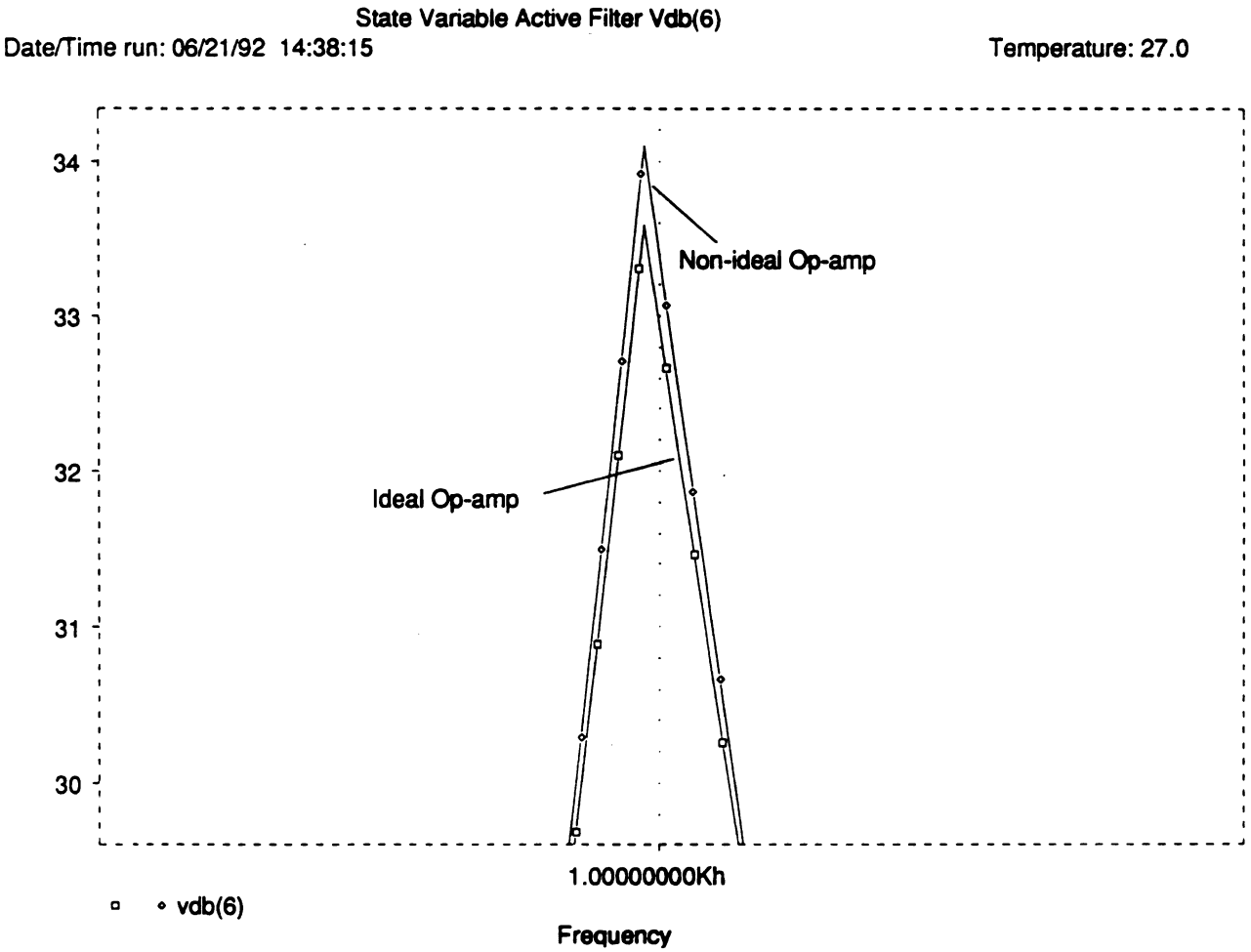


Figure 7.8. Pspice simulation of State Variable Active Filter using ideal and nonideal op-amps.

SEN(arg1,arg2) Perform sensitivity analysis of **arg1** with respect to **arg2** by using *Sensitivity Algebra.* Usually, **arg1** is an expression, and **arg2** is an element.

QSN(arg1,arg2) Perform quick sensitivity analysis of **arg1** with respect to **arg2** by using *Quick sensitivity algorithm* which will be described in Section III.

DIF(arg1,arg2) Perform the first derivative of **arg1** with respect to **arg2**.

TRM(arg1,arg2) Extract the coefficient of s's **arg2**'th order terms of **arg1**'s numerator.

NUM(arg) Extract the numerator of **arg**.

DEN(arg) Extract the denominator of **arg**

SMY(arg) Perform numerical approximation and numerator-denominator common factor elimination.

HTZ(arg) Perform Hurwitz Test to the polynomial **arg**.

QO2(arg) Extract the square of the quality factor of a second order filter function, **arg**.

WO2(arg) Extract the square of the center frequency of a second order filter function, **arg**.

ZO2(arg) Extract the square of the bandstop frequency of a second order filter function, **arg**.

Example 17 *The sensitivity of V6 with respect to C1 can be obtained by commanding* **SEN(V6,C1)**. *If numerical approximation is specified, the* **QSN(V6,C1)** *command can further improve the computation efficiency.* **SEN(QO2(V6),C1)/2** *would give Q sensitivity with respect to C1.*

Example 18 *The stability analysis of V6 can be done by doing Hurwitz Test to the denominator of V6. This is achieved by using* `HTZ(DEN(V6))`.

CHAPTER 8

CONCLUSIONS

In this dissertation, the development and the applications of a symbolic analog circuit analyzer are presented. It has been understood that the use of symbolic circuit analyzer is not a one time solution. A circuit designer has to do trial and error many times in order to obtain the solutions he/she needs. This requires extensive experience of using symbolic tools and in-depth understandings of the circuits. Adapting a symbolic circuit analyzer into the design platform with an appropriate education certainly would improve both of the above factors. On the other hand, a symbolic circuit analyzer should be made as flexible as possible so that circuit designers can fully utilize their imagination and knowledge without restriction. Sometimes, this may produce confusion among different interpretations of the results. Users have to understand the meanings of the results precisely so that the possibility of getting a faulty conclusion can be minimized. Besides efficiency, therefore, flexibility and interpretability are also the major concerns while developing Sspice version 2.0.

In Chapter 3, a new strategy for circuit composition and decomposition approach was presented. Before, decomposition strategies based on graph algorithms didn't provide a unified method to obtain the network determinant of a whole circuit from the decomposed subcircuits for both passive and active circuits. The presented new strategy alleviats this difficulty. Also, this strategy improves both the computation

efficiency and memory consumption. It is hard to develop an unified method to find an optimum cut set for all the possible circuits. However, the analysis provided in Section 3.3.1 gives the hints of the possible solutions. The follow-up theorems in Section 3.4 shows that there are special cases that can be taken advantage of to further improve the efficiency. Section 3.5 shows an example which reduces a 20 case decomposition approach into only 2 cases.

In Chapter 4, the approximation techniques implemented in Sspice are presented. For small circuits of less than 20 components, numerical approximation after computation technique usually is good enough. However, for larger circuits, numerical approximation during computation and before computation techniques are needed. Example 15 shows the combination of decomposition and approximation before computation to solve a well known problem.

The symbolic sensitivity analysis and symbolic stability analysis in Chapter 5 and Chapter 6 explore new applications by providing suitable software tools. The examples in these chapters show that symbolic approach provides a better solution in comparison with the numerical approach.

Chapter 7 explains implementation of Sspice version 2.0 including the matrix reduction strategy. Sspice version 2.0 is a symbolic circuit analyzer, which is compatible with version 1.0, with extensive capabilities. It is no longer a pure symbolic circuit analyzer. Its internal data structures are now capable of handling both numeric and symbolic information. This provides more flexibility and efficiency for solving circuit problems. Sspice version 2.0 has been tested and applied to the development of macromodels of power supply regulators [28].

The future of the symbolic approach depends on the integration of the symbolic analyzer into the existing numerical based design process and system. Then, circuit designers would have more opportunities for accessing symbolic analysis tools and get better educated in using them appropriately. By applying symbolic approach into a

design project, a circuit designer would find more needs and new applications of symbolic analysis. The needs for symbolic sensitivity analysis and symbolic Hurwitz test, for example, were discovered by circuit designers on their jobs.

Symbolic analog circuit analysis is now basically applied to the nodal analysis platform. It is for linear circuits. There are other research opportunities which may be suitable for symbolic approaches for nonlinear circuits. Translinear analysis could be the next frontier for symbolic analog circuit analysis.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Ismail and S. Bibyk, "Cad latches onto new technieques for analog ics," *IEEE Circuits and Devices Magazine*, pp. 11 - 17, Sept. 1991.

[2] P. Lin, "A survey of applications of symbolic network functions," *IEEE Transactions on Circuits and Systems*, pp. 732 - 737, Nov. 1973.

[3] J. Vlach, *Computer Methods for Circuit Analysis and Design.* Van Nostrand Reinhold Company, 1983.

[4] G. Wierzba *et al.*, "Sspice - a symbolic spice program for linear active circuits," in *Proc. 32nd Midwest Symp on CAS*, pp. 1197 - 1201, 1989.

[5] G. Gielen, H. Walscharts, and W. Sansen, "Isaac: A symbolic simulator for analog integrated circuits," *IEEE Journal of Solid-State Circuits*, pp. 1587 - 1597, Dec. 1989.

[6] H. Carlin and D. Youla, "Network synthesis with negative resistors," *Proceedings of The IRE*, pp. 907 - 920, May 1961.

[7] S. K. Mitra, *Analysis and Synthesis of Linear Active Networks.* John Wiley and Sons, Inc., 1969.

[8] W.-K. Chen, "Topological analysis for active networks," *IEEE Transactions on Circuits and Systems*, pp. 85 - 91, Jan. 1965.

[9] W.-K. Chen, *Applied Graph Theory.* North-Holland Publishing Company, 1976.

[10] M. Hassoun, "Hierarchical symbolic analysis of large-scale systems using a mason's signal flow graph model," in *Proc. ISCAS*, pp. 802 - 805, 1991.

[11] J. A. Starzyk and A. Konczykowska, "Flowgraph analysis of large electronic networks," *IEEE Transactions on Circuits and Systems*, pp. 302 - 315, Mar. 1986.

[12] T. Ozawz, ed., *Analog Methods for Computer-Aided Circuit Analysis and Diagnosis.* Marcel Dekker, Inc., 1988.

[13] L. O. Chua and P.-M. Lin, *Computer-Aided Analysis of Electronic Circuits.* Prentice-Hall, Inc., 1975.

[14] P. Lin and G. Alderson, *SNAP - A Computer Program for Generating Symbolic Network Functions.* School of Elec. Engr., Purdue Univ., Lafayette, Indiana, Aug. 1970.

[15] L. Huelsman, "Personal computer symbolic analysis programs for undergraduate engineering courses," in *Proc. ISCAS*, pp. 798 - 801, 1989.

[16] M. Degrauwe *et al.*, "Idac: An interactive design tool for analog cmos circuits," *IEEE Journal of Solid-State Circuits*, Dec. 1987.

[17] R. Harjani *et al.*, "A prototype framework for knowledge based analog circuit symthesis," in *Proc. IEEE Design Automation Conference*, pp. 42 - 49, 1987.

[18] H. Koh *et al.*, "Automatic synthesis of operational amplifiers based on analytic circuit models," in *Proc. ICCAD*, pp. 502 - 505, 1987.

[19] S. Seda *et al.*, "A symbolic analysis tool for analog circuit design automation," in *Proc. IEEE ICCD*, pp. 488 - 491, 1988.

[20] J. Deutsch and A. Newton, "A multiprocessor implementation of relaxation-based electrical circuit simulation," in *Proc. Design Automation Conference*, pp. 350 - 357, 1984.

[21] J. Starzyk and E. Sliwa, "Upward topological analysis of large circuits using directed graph representation," *IEEE Transactions on Circuits and Systems*, pp. 410 - 414, Apr. 1984.

[22] M. Hassoun and P. Lin, "A new network approach to symbolic simulation of large-scale networks," in *Proc. ISCAS*, pp. 806 - 809, 1989.

[23] H. Walscharts, G. Gielen, and W. Sansen, "Symbolic simulation of analog circuits in s- and z-domain," in *Proc. ISCAS*, pp. 814 - 817, 1989.

[24] L. Huelsman and D. Dalton, "A computational approach to the development and reduction of large symbolic expressions," in *Proc. ISCAS*, pp. 790 - 793, 1991.

[25] G. M. Wierzba, *Sspice User Manual.* Instructional Media Center, Michigan State University, 1991.

[26] E. A. Guillemin, *Synthesis of Passive Networks.* Chapman and Hall, limited, 1957.

[27] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic approximation strategies and the symbolic analysis of large and nonlinear circuits," in *Proc. ISCAS*, pp. 806 - 809, 1991.

[28] K. Noren, *Macromodeling of Voltage Regulator IC's.* Dept. of Elec. Engr., Michigan State Univ., E. Lansing, Michigan, July 1992.