8

Symbolic Analysis

Benedykt S. Rodanski
University of Technology, Sydney

Marwan M. Hassoun
Iowa State University

8.1 Introduction and Definition

Symbolic circuit analysis, simply stated, is a term that describes the process of studying the behavior of electrical circuits using symbols instead of, or in conjunction with, numerical values. As an example to illustrate the concept, consider the input resistance of the simple circuit in Figure 8.1. Analyzing t**8**-he circuit using the unique symbols for each resistor without assigning any numerical values to them yields the input resistance of the circuit in the form:

$$\frac{V_{in}}{I_{in}} = \frac{R_1R_2 + R_1R_3 + R_1R_4 + R_2R_3 + R_2R_4}{R_2 + R_3 + R_4} \tag{8.1}$$

Equation (8.1) is the symbolic expression for the input resistance of the circuit in Figure 8.1.

The formal definition of symbolic circuit analysis can be written as:

Definition 1. Symbolic circuit analysis is the process of producing an expression that describes a certain behavioral aspect of the circuit with one, some, or all the circuit elements represented as symbols.

The idea of symbolic circuit analysis is not new; engineers and scientists have been using the process to study circuits since the inception of the concept of circuits. Every engineer has used symbolic circuit analysis during his or her education process. Most engineers still use it in their everyday job functions. As an example, all electrical engineers have symbolically analyzed the circuit in Figure 8.2. The equivalent resistance between nodes i and j is known to be:

$$\frac{1}{R_{ij}} = \frac{1}{R_1} + \frac{1}{R_2}$$

or

$$R_{ij} = \frac{R_1R_2}{R_1 + R_2}$$

This is the most primitive form of symbolic circuit analysis.

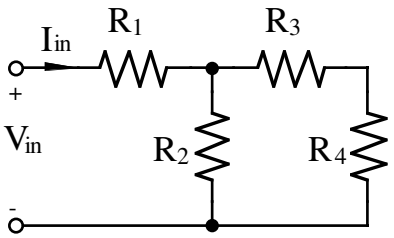

FIGURE 8.1 Symbolic circuit analysis example.

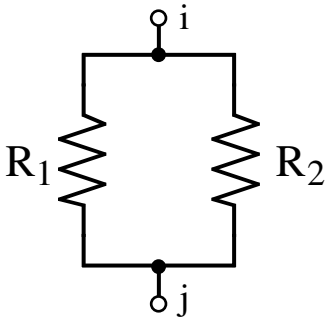

FIGURE 8.2 Common symbolic analysis problem.

The basic justification for performing symbolic analysis rather than numerical analysis on a circuit can be illustrated by considering the circuit in Figure 8.1 again. Assume that the values of all the resistances R_1 through R_4 are given as 1Ω and that the input resistance was analyzed numerically. The result obtained would be

$$\frac{V_{in}}{I_{in}} = \frac{5}{3} \approx 1.667\ \Omega \tag{8.2}$$

Now, consider the problem of increasing the input resistance of the circuit by adjusting only one of the resistor values. Equation (8.2) provides no insight into which resistor has the greatest impact on the input resistance. However, Eq. (8.1) clearly demonstrates that changing R_2, R_3, or R_4 would have very little impact on the input resistance because the terms appear in both the numerator and the denominator of the symbolic expression. It can also be observed that R_1 should be the resistor to change because it only appears in the numerator of the expression. Symbolic analysis has provided an insight into the problem.

From a circuit design perspective, numerical results from the simulation of a circuit can be obtained by evaluating the results of the symbolic analysis at a specific numerical point for each symbol. Ideally, only one simulation run is needed in order to analyze the circuit, and successive evaluations of the results replaces the need for any extra iterations through the simulator. Other applications include sensitivity analysis, circuit stability analysis, device modeling and circuit optimization [5, 18, 32, 41].

Although the previous "hand calculations" and somewhat trivial examples are used to illustrate symbolic circuit analysis, the thrust of the methods developed for symbolic analysis are aimed at computer implementations that are capable of symbolically analyzing circuits that cannot be analyzed "by hand." Several such implementations have been developed over the years [4, 10, 12, 15, 17, 22, 25, 26, 28, 29, 31, 34, 35, 37, 38, 47, 48, 53, 54, 56–61, 66].

Symbolic circuit analysis, referred to simply as symbolic analysis for the rest of this section, in its current form is limited to linear,[1] lumped, and time-invariant[2] networks. The scope of the analysis is

[1]Some references are made to the ability to analyze "weakly nonlinear" circuits [18, 63]; however, the actual symbolic analysis is performed on a linearized model of the weakly nonlinear circuit. Other techniques are applicable to circuits with only a single strongly nonlinear variable [65].

[2]One method is reported in Reference [36] that is briefly discussed in Section 8.6 that does deal with a limited class of time-variant networks.

primarily concentrated in the frequency domain, both *s*-domain [10, 12, 15, 17, 22, 25, 28, 29, 34, 38, 43, 47, 48, 54, 56, 59–61, 66] and *z*-domain [4, 31, 35, 44]; however, the predominant development has been in the *s*-domain. Also, recent work has expanded symbolic analysis into the time domain [3, 24, 36]. The next few subsections will discuss the basic methods used in symbolic analysis for mainly *s*-domain frequency analysis. However, Section 8.6 highlights the currently known time-domain techniques.

8.2 Frequency-Domain Analysis

Traditional symbolic circuit analysis is performed in the frequency domain where the results are in terms of the frequency variable *s*. The main goal of performing symbolic analysis on a circuit in the frequency domain is to obtain a symbolic transfer function of the form

$$H(s,\mathbf{x}) = \frac{N(s,\mathbf{x})}{D(s,\mathbf{x})}, \quad \mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix}, \quad p \le p_{all} \tag{8.3}$$

The expression is a rational function of the complex frequency variable s, and the variables x_1 through x_p representing the variable circuit elements, where p is the number of variable circuit elements and p_{all} is the total number of circuit elements. Both the numerator and the denominator of $H(s,\mathbf{x})$ are polynomials in s with real coefficients. Therefore, we can write

$$H(s,\mathbf{x}) = \frac{\sum_{i=0}^{m} a_i(\mathbf{x}) s^i}{\sum_{i=0}^{n} b_i(\mathbf{x}) s^i} = \frac{\prod_{i=1}^{m} \left[s - z_i(\mathbf{x}) \right]}{\prod_{i=1}^{n} \left[s - p_i(\mathbf{x}) \right]}$$

Most symbolic methods to date concentrate on the first form of $H(s,\mathbf{x})$ and several algorithms exist to obtain coefficients $a_i(\mathbf{x})$ and $b_i(\mathbf{x})$ in fully symbolic, partially symbolic (semi-symbolic), or numerical form. The zero/pole representation of $H(s,\mathbf{x})$, although more useful in gaining insight into circuit behavior, proved to be very difficult to obtain in symbolic form for anything but very simple circuits. For large circuits, various approximation techniques must be employed [9, 26].

A more recent approach to representing the above network function emerged in the 1980s and is based on a decomposed hierarchical form of Eq. (8.3) [22, 25, 51, 61, 62]. This hierarchical representation is referred to as a *sequence of expressions* representation to distinguish it from the *single expression* representation of Eq. (8.3) and is addressed in Section 8.4.

Several methodologies exist to perform symbolic analysis in the frequency domain. The early work was to produce a transfer function $H(s)$ with the frequency variable s being the only symbolic variable. Computer programs with these capabilities include: CORNAP [54] and NASAP [47]. The interest in symbolic analysis today is in the more general case when some or all of the circuit elements are represented by symbolic variables. The methods developed for this type of analysis fall under one of the following categories:

Traditional methods (single expression):

1. Tree enumeration methods
   - Single graph methods
   - Two graph methods
2. Signal flow graph methods
3. Parameter extraction methods
   - Modified nodal analysis-based methods
   - Tableau formulation-based methods
4. Interpolation method

Hierarchical methods (sequence of expressions):

1. Signal flow graph methods
2. Modified nodal analysis-based methods

The preceding classification includes the exact methods only. For large circuits, the traditional methods suffer from exponential growth of the number of terms in the formula with circuit size. If a certain degree of error is allowed, it may be possible to simplify the expression considerably, by including only the most significant terms. Several *approximate symbolic methods* have been investigated [26, 28, 69].

The next three sections discuss the basic theory for the above methods. Circuit examples are illustrated for all major methods except for the interpolation method due to its limited current usage[3] and its inability to analyze fully symbolic circuits.

8.3 Traditional Methods (Single Expressions)

This class of methods attempts to produce a single transfer function in the form of Eq. (8.3). The major advantage of having a symbolic expression in that form is the insight that can be gained by observing the terms in both the numerator and the denominator. The effects of the different terms can, perhaps, be determined by inspection. This process is valid for the cases where relatively few symbolic terms are in the expression.

Before indulging in the explanation of the different methods covered by this class, some definition of terms is in order.

Definition 2. *$RLCg_m$ circuit* is one that may contain only resistors, inductors, capacitors, and voltage-controlled current sources with the gain (transconductance) designated as g_m.

Definition 3. *Term cancellations* is the process in which two equal symbolic terms cancel out each other in the symbolic expression. This can happen in one of two ways: by having two equal terms with opposite signs added together, or by having two equal terms (regardless of their signs) divided by each other. For example, the equation

$$\frac{ab(ab+cd)-ab(cd-ef)}{ab(cd-gh)} \tag{8.4}$$

where a, b, c, d, e, f, g, and h are symbolic terms, can be reduced by observing that the terms ab in the numerator and denominator cancel each other and the terms $+cd$ and $-cd$ cancel each other in the numerator. The result is:

$$\frac{ab+ef}{cd-gh} \tag{8.5}$$

Definition 4. *Cancellation-free:* Equation (8.4) is said to be a cancellation-free equation (that is, no possible cancellations exist in the expression) while Eq. (8.5) is not.

Definition 5. *Cancellation-free algorithm*: The process of term cancellation can occur during the execution of an algorithm where a cancellation-free equation is generated directly instead of generating an expression with possible term cancellations in it. Cancellation-free algorithms are more desirable because, otherwise, an overhead is needed to generate and keep the terms that are to be canceled later.

[3]The main applications of the polynomial interpolation method in symbolic analysis are currently in numerical reference generation for symbolic approximation [14] and calculation of numerical coefficients in semi-symbolic analysis [50].

The different methods that fall under the traditional class are explained next.

1. The tree enumeration methods

Several programs have been produced based on this method [6, 16, 42, 46]. Practical implementations of the method can only handle small circuits in the range of 15 nodes and 30 branches [7]. The main reason is the exponential growth in the number of symbolic terms generated. The method can only handle one type of controlled source, namely, voltage controlled current sources. So only RLCg_m circuits can be analyzed. Also, the method does not produce any symbolic term cancellations for RLC circuits, and produces only a few for RLCg_m circuits.

The basic idea of the tree enumeration method is to construct an augmented circuit (a slightly modified version of the original circuit), its associated directed graph, and then enumerating all the directed trees of the graph. The admittance products of these trees are then used to find the node admittance matrix determinant and cofactors (the matrix itself is never constructed) to produce the required symbolic transfer functions. For a circuit with n nodes (with node n designated as the reference node) where the input is an excitation between nodes 1 and n and the output is taken between nodes 2 and n, the transfer functions of the circuit can be written as:

$$Z_{in} = \frac{V_1}{I_1} = \frac{\Delta_{11}}{\Delta} \tag{8.6}$$

$$\frac{V_o}{I_{in}} = \frac{V_2}{I_1} = \frac{\Delta_{12}}{\Delta} \tag{8.7}$$

$$\frac{V_o}{V_{in}} = \frac{V_2}{V_1} = \frac{\Delta_{12}}{\Delta_{11}} \tag{8.8}$$

where Δ is the determinant of the node admittance matrix $\mathbf{Y}_n$ (dimension n-1 × n-1) and Δ_{ij} is the ijth cofactor of $\mathbf{Y}_n$. It can be shown that a simple method for obtaining Δ, Δ_{11}, and Δ_{12} is to construct another circuit comprised of the original circuit with an extra admittance $\hat{y}_s$ in parallel with a voltage controlled current source, $\hat{g}_m V_2$, connected across the input terminals (nodes 1 and n). The determinant of $\hat{\mathbf{Y}}_n$ (the node admittance matrix for the new, slightly modified, circuit) can be written as:

$$\hat{\Delta} = \Delta + \hat{y}_S \Delta_{11} + \hat{g}_m \Delta_{12} \tag{8.9}$$

This simple trick allows the construction of the determinant expression of the original circuit and its two needed cofactors by simply formulating the expression for the new augmented circuit. Example 8.1 below illustrates this process.

The basic steps of the tree enumeration algorithm are (condensed from [7]):

1. Construct the augmented circuit from the original circuit by adding an admittance $\hat{y}_s$ and a transconductance $\hat{g}_m V_2$, in parallel between the input node and the reference node.
2. Construct a directed graph $\mathbf{G}_{\text{ind}}$ associated with the augmented circuit. The stamps used to generate $\mathbf{G}_{\text{ind}}$ are illustrated in Figure 8.3.
3. Find all directed trees for $\mathbf{G}_{\text{ind}}$. A directed tree rooted at node i is a subgraph of $\mathbf{G}_{\text{ind}}$ with node i having no incoming branches and each other node having exactly one incoming branch.
4. Find the admittance product for each directed tree. An admittance product of a directed tree is simply a term that is the product of all the weights of the branches in that tree.
5. Apply the following theorem:

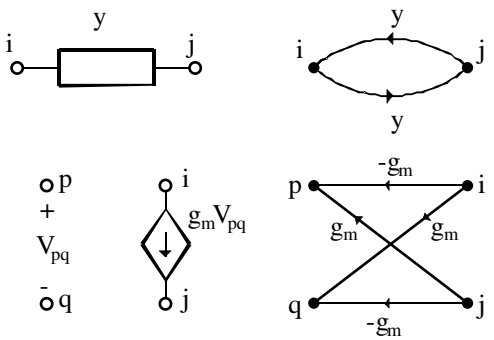

FIGURE 8.3 Element stamps for generating $\mathbf{G}_{ind}$.

Theorem 8.1 [7]: For any RLCg_m circuit, the determinant of the node admittance matrix (with any node as the reference node) is equal to the sum of all directed tree admittance products of $\mathbf{G}_{ind}$ (with any node as the root).

In other words

$$\hat{\Delta} = \sum \text{tree admittance products} \tag{8.10}$$

Arranging Eq. (8.10) in the form of Eq. (8.9) results in the necessary determinant and cofactors of the original circuit and the required transfer functions are generated from Eqs. (8.6), (8.7), and (8.8).

Example 1. A circuit and its augmented counterpart are illustrated in Figure 8.4. The circuit is the small-signal model of a simple inverting CMOS amplifier, shown with the coupling capacitance C_C taken into account. Figure 8.5 depicts the directed graph associated with the augmented circuit constructed using the rules in Figure 8.3. The figure also presents all the directed trees rooted at node 3 of the graph. Parallel branches heading in the same direction are combined into one branch with a weight equal to the sum of the weights of the individual parallel branches.

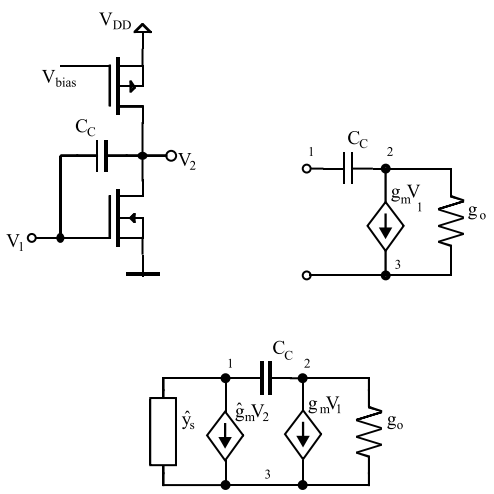

FIGURE 8.4 Circuit of Example 8.1 and its augmented equivalent diagram.

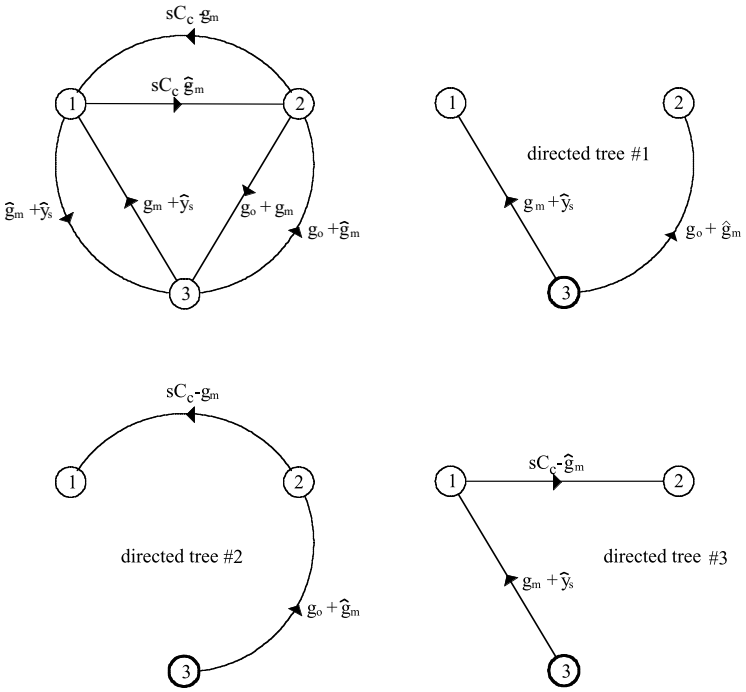

FIGURE 8.5 Graph and its directed trees of Example 8.1.

Applying Eq. (8.10) and rearranging the terms results in:

$$\begin{aligned}\hat{\Delta} &= \left(g_m + \hat{y}_S\right)\left(g_o + \hat{g}_m\right) + \left(sC_C - g_m\right)\left(g_o + \hat{g}_m\right) + \left(g_m + \hat{y}_S\right)\left(sC_C - \hat{g}_m\right) \\ &= \underbrace{sC_C\left(g_m + g_o\right)}_{\Delta} + \hat{y}_S\underbrace{\left(sC_C - g_o\right)}_{\Delta_{11}} + \hat{g}_m\underbrace{\left(sC_C - g_m\right)}_{\Delta_{12}}\end{aligned} \tag{8.11}$$

Note the fact that Eq. (8.11), which is the direct result of the algorithm, is not cancellation-free. Some terms cancel out to result in the determinant of the original circuit and its two cofactors of interest. The final transfer functions can be obtained readily by substituting the preceding results into Eq. (8.6) through (8.8).

2. The signal flow graph method

Two types of flow graphs are used in symbolic analysis. The first is referred to as a Mason's SFG and the second as Coates graph. Mason's SFG is by far a more popular and well-known SFG that has been used extensively in symbolic analysis among other controls applications. Both the Mason's SFG and the Coates graph are used as a basis for hierarchical symbolic analysis. However, the Coates graph was introduced to symbolic analysis by Starzyk and Konczykowska [61] solely for the purpose of performing hierarchical symbolic analysis. This section covers the Mason's SFG only.

The symbolic methods developed here are based on the idea formalized by Mason [45] in the 1950s. Formulation of the signal flowgraph and then the evaluation of the gain formula associated with it (Mason's formula) is the basis for symbolic analysis using this method. This method is used in the publicly available programs NASAP [47, 49] and SNAP [38]. The method has the same circuit size limitations as the tree enumeration method due to the exponential growth in the number of symbolic terms. However,

the signal flowgraph method allows all four types of controlled sources to be analyzed which made it a more popular method for symbolic analysis. The method is not cancellation-free, which contributes to the circuit size limitation mentioned earlier. An improved signal flowgraph method that avoids term cancellations was described in [48].

The analysis process of a circuit consists of two parts: the first is constructing the SFG for the given circuit and the second is to perform the analysis on the SFG. Some definitions are needed before proceeding to the details of these two parts.

Definition 6. *Signal Flow Graph:* An SFG is a weighted directed graph representing a system of simultaneous linear equations. Each node (x_i) in the SFG represents a circuit variable (node voltage, branch voltage, branch current, capacitor charge, or inductor flux) and each branch weight (w_{ij}) represents a coefficient relating x_i to x_j.

Every node in the SFG can be looked at as a summer. For a node x_k with m incoming branches

$$x_k = \sum_i w_{ik} x_i \tag{8.12}$$

where i spans the indices of all incoming branches from x_i to x_k.

Definition 7. *Path Weight:* The weight of a path from x_i to x_j (P_{ij}) is the product of all the branch weights in the path.

Definition 8. *Loop Weight:* The weight of a loop is the product of all the branch weights in that loop. This also holds for a loop with only one branch in it (self-loop).

Definition 9. *nth Order Loop:* An nth order loop is a set of n loops that have no common nodes between any two of them. The weight of an nth order loop is the product of the weights of all n loops.

Any transfer function x_j/x_i, where x_i is a source node, can be found by the application of Mason's formula:

$$\frac{x_j}{x_i} = \frac{1}{\Delta} \sum_k P_k \Delta_k \tag{8.13}$$

where

$$\begin{aligned} \Delta = 1 &- \sum_{\text{all}} \text{directed loop weights} \\ &+ \sum_{all} \text{2nd-order loop weights} \\ &- \sum_{all} \text{3rd-order loop weights} \\ &+ \cdots \end{aligned} \tag{8.14}$$

$$\begin{aligned} P_k &= \text{weight of the } k\text{th path from the source node } x_i \text{ to } x_j \\ \Delta_k &= \Delta \text{ with all loop contributions that are touching } P_k \text{ eliminated} \end{aligned} \tag{8.15}$$

The use of the preceding equations can be illustrated via an example.

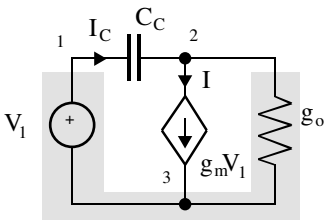

FIGURE 8.6 Circuit for Example 8.2 with its tree highlighted.

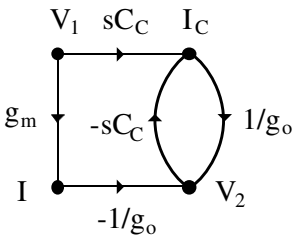

FIGURE 8.7 SFG for Example 8.2.

Example 2. Consider the circuit in Figure 8.6. The formulation of the SFG for this circuit takes on the following steps:

1. Find a tree and a co-tree of the circuit such that all current sources are in the co-tree and all voltage sources are in the tree.
2. Use Kirchhoff's current law (KCL), branch admittances, and tree branch voltages to find an expression for every co-tree link current. In the case of a controlled source, simply use the branch relationship. For the previous example, this yields:

$$I_C = sC_C\left(V_1 - V_2\right) = sC_CV_1 - sC_CV_2$$

$$I = g_mV_1$$

3. Use Kirchhoff's voltage law (KVL), branch impedances, and co-tree link currents to find an expression for every tree branch voltage. In the case of a controlled source, simply use the branch relationship. For the previous example, this yields:

$$V_{g_o} = V_2 = \frac{1}{g_o}\left(-I + I_C\right)$$

4. Create the SFG by drawing a node for each current source, voltage source, tree branch voltage, and co-tree link current.
5. Use Eq. (8.12) to draw the branches between the nodes that realize the linear equations developed in the previous steps.

Figure 8.7 is the result of executing the preceding steps on the example circuit. This formulation is referred to as the compact SFG. Any other variables that are linear combinations of the variables in the SFG (e.g., node voltages) can be added to the SFG by simply adding the extra node and implementing the linear relationship using SFG branches. A more detailed discussion of SFGs can be found in [7] and [40].

Now applying Eqs. (8.14) and (8.15) yields:

$$P_1 = -\frac{g_m}{g_o},\quad P_2 = \frac{sC_C}{g_o},\quad L_1 = -\frac{sC_C}{g_o},\quad \Delta = 1 - \left(-\frac{sC_C}{g_o}\right),\quad \Delta_1 = 1,\quad \Delta_2 = 1$$

Equation (8.13) then produces the final transfer function

$$\frac{V_2}{V_1} = \frac{1}{1+\dfrac{sC_C}{g_o}}\left(-\frac{g_m}{g_o}+\frac{sC_C}{g_o}\right) = \frac{sC_C - g_m}{sC_C + g_o}$$

3. The parameter extraction method

This method is best suited when few parameters in a circuit are symbolic while the rest of the parameters are in numeric form (*s* being one of the symbolic variables). The method was introduced in 1973 [2]. Other variations on the method were proposed later in [50, 56, 59]. The advantage of the method is that it is directly related to the basic determinant properties of widely used equation formulation methods such as the modified nodal method [27] and the tableau method [21]. As the name of the method implies, it provides a mechanism for extracting the symbolic parameters out of the matrix formulation, breaking the matrix solution problem into a numeric part and a symbolic part. The numeric part can then be solved using any number of standard techniques and recombined with the extracted symbolic part. The method has the advantage of being able to handle larger circuits than the previously discussed fully symbolic methods if only a few parameters are represented symbolically. If the number of symbolic parameters in a circuit is high, the method will exhibit the same exponential growth in the number of symbolic terms generated and will have the same circuit size limitations as the other algorithms previously discussed.

The method does not limit the type of matrix formulation used to analyze the circuit. However, the extraction rules depend on the pattern of the symbolic parameters in the matrix. Alderson and Lin [1] use the indefinite admittance matrix as the basis of the analysis and the rules depend on the appearance of a symbolic parameter in four locations in the matrix: (*i*,*i*), (*i*,*j*), (*j*,*i*), and (*j*,*j*). Singhal and Vlach [59] use the tableau equations and can handle a symbolic parameter that only appears once in the matrix. Sannuti and Puri [56] force the symbolic parameters to appear only on the diagonal using a two-graph method [7] to write the tableau equations. The parameter extraction method was further simplified in [50], where the formula is given to calculate a coefficient (generally a polynomial in *s*) at every symbol combination. Some invalid symbol combinations (i.e., the ones that do not appear in the final formula) can be eliminated before calculations by topological considerations. To illustrate both approaches to parameter extraction, this section presents the indefinite admittance matrix (IAM) formulation and the most recent two-graph method. Details of other formulations can be found in [40, 48, 56, 59].

Indefinite Admittance Matrix Approach

One of the basic properties of the IAM is the symmetric nature of the entries sometimes referred to as *quadrantal* entries [7, 40]. A symbolic variable α will always appear in four places in the indefinite admittance matrix, $+\alpha$ in entries (i, k) and (j, m), and $-\alpha$ in entries (i, m) and (j, k) as demonstrated in the following equation:

$$\begin{array}{c} \\ i \\ \\ j \\ \\ \end{array}\begin{array}{c} \begin{array}{ccc} k & & m \end{array} \\ \begin{bmatrix} \vdots & & \vdots \\ \alpha & \cdots & -\alpha \\ \vdots & & \vdots \\ -\alpha & \cdots & -\alpha \\ \vdots & & \vdots \end{bmatrix} \end{array}$$

where $i \neq j$ and $k \neq m$. For the case of an admittance y between nodes i and j, we have $k = i$ and $j = m$. The basic process of extracting the parameter (the symbol) α can be performed by applying the following equation [2, 7]:

$$\text{cofactor of } \mathbf{Y}_{ind} = \text{cofactor of } \mathbf{Y}_{ind,\alpha=0} + (-1)^{j+m}\alpha\left(\text{cofactor of } \mathbf{Y}_{\alpha}\right) \tag{8.16}$$

where $\mathbf{Y}_{\alpha}$ is a matrix that does not contain α and is obtained by:

1. Adding row j to row i
2. Adding column m to column k
3. Deleting row j and column m

For the case where several symbols exist, the previous extraction process can be repeated and would result in

$$\text{cof}\left(\mathbf{Y}_{ind}\right) = \sum_{j} P_j \,\text{cof}\left(\mathbf{Y}_j\right)$$

where P_j is some product of symbolic parameters including the sign and $\mathbf{Y}_j$ is a matrix with the frequency variable s, possibly being the only symbolic variable. The cofactor of $\mathbf{Y}_j$ may be evaluated using any of the usual evaluation methods [7, 64]. Programs implementing this technique include NAPPE2 [40] and SAPWIN [37].

Example 4 [7]. Consider the resistive circuit in Figure 8.8. The goal is to find the input impedance Z_{14} using the parameter extraction method where g_m is the only symbolic variable in the circuit. In order to use Eqs. (8.6) and (8.9), an admittance $\hat{y}_s$ is added across the input terminals of the circuit to create the augmented circuit.

The IAM is then written as (conductances in siemens [S])

$$\hat{\mathbf{Y}}_{ind} = \begin{bmatrix} 6+\hat{y}_S & -5 & -1 & -\hat{y}_S \\ g_m - 5 & 15.1 & -g_m - 10 & -0.1 \\ -g_m - 1 & -10 & g_m + 13 & -2 \\ -\hat{y}_S & -0.1 & -2 & \hat{y}_S + 2.1 \end{bmatrix}$$

Applying Eq. (8.16) to extract $\hat{y}_s$ results in

$$\text{cof}\left(\hat{\mathbf{Y}}_{ind}\right) = \text{cof}\begin{bmatrix} 6 & -5 & -1 & 0 \\ g_m - 5 & 15.1 & -g_m - 10 & -0.1 \\ -g_m - 1 & -10 & g_m + 13 & -2 \\ 0 & -0.1 & -2 & 2.1 \end{bmatrix} + \hat{y}_S \,\text{cof}\begin{bmatrix} 8.1 & -5.1 & -3 \\ g_m - 5.1 & 15.1 & -g_m - 10 \\ -g_m - 3 & -10 & g_m + 13 \end{bmatrix}$$

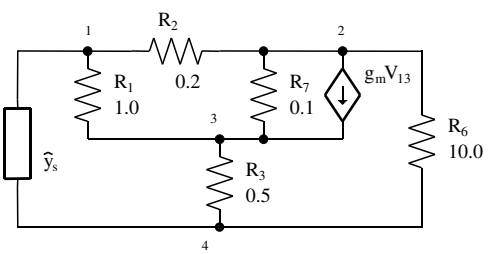

FIGURE 8.8 Circuit for the parameter extraction method (resistances in ohms [Ω]).

Applying Eq. (8.16) again to extract g_m yields

$$\operatorname{cof}\left(\hat{\mathbf{Y}}_{ind}\right) = \operatorname{cof}\begin{bmatrix} 6 & -5 & -1 & 0 \\ -5 & 15.1 & -10 & -0.1 \\ -1 & -10 & +13 & -2 \\ 0 & -0.1 & -2 & 2.1 \end{bmatrix} + g_m \operatorname{cof}\begin{bmatrix} 5 & -5 & 0 \\ -3 & 5.1 & -2.1 \\ -2 & -0.1 & 2.1 \end{bmatrix}$$

$$+\hat{y}_S \operatorname{cof}\begin{bmatrix} 8.1 & -5.1 & -3 \\ -5.1 & 15.1 & -10 \\ -3 & -10 & 13 \end{bmatrix} + \hat{y}_S g_m \operatorname{cof}\begin{bmatrix} 5.1 & -5.1 \\ -5.1 & 5.1 \end{bmatrix}$$

After evaluating the cofactors numerically, the equation reduces to

$$\operatorname{cof}\left(\hat{\mathbf{Y}}_{ind}\right) = 137.7 + 10.5g_m + 96.3\hat{y}_S + 5.1\hat{y}_S g_m$$

From Eq. (8.9), this results in

$$Z_{14} = \frac{\Delta_{11}}{\Delta} = \frac{96.3 + 5.1g_m}{137.7 + 10.5g_m}$$

Two-Graph-Based Tableau Approach [50]

This approach also employs the circuit augmentation by $\hat{y}_s$ and $\hat{g}_m V_o$, as in the tree enumeration method. It calls for the construction of two graphs: the voltage graph (G_V or V-graph) and the current graph (G_I or I-graph). For the purpose of parameter extraction (as well as generation of approximate symbolic expressions; see Section 8.5), it is required that both graphs have the same number of nodes (n). This means that the method can be directly applied only to RLCg_m circuits. (All basic circuit components, including ideal op amps, can be handled by this approach after some circuit transformations [40, 64]. For the sake of simplicity, however, only RLCg_m circuits will be considered in this presentation.) The two graphs are constructed based on the element stamps shown in Figure 8.9. Once the two graphs are constructed, a *common spanning tree* (i.e., a set of n-1 branches that form a spanning tree in both voltage and current graphs) is chosen. Choosing the common spanning tree (referred to just as "tree" in the remainder of this section) uniquely determines the co-tree in each graph.

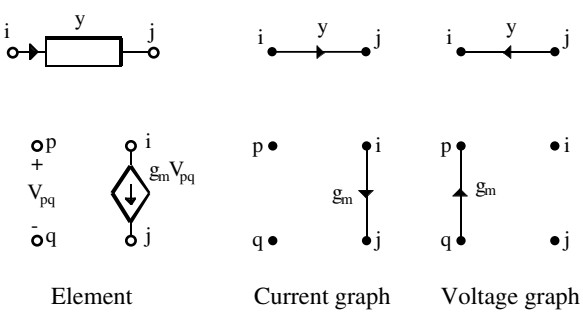

FIGURE 8.9 Element stamps for generating G_V and G_I.

The tableau equation for such a network can be written as

$$\mathbf{Hx} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & -\mathbf{Z}_T & \mathbf{0} \\ \mathbf{B}_T & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{Q}_C \\ \mathbf{0} & -\mathbf{Y}_C & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{V}_T \\ \mathbf{V}_C \\ \mathbf{I}_T \\ \mathbf{I}_C \end{bmatrix} = \mathbf{0} \tag{8.17}$$

The first and last row of the system matrix **H** in Eq. (8.17) consists of tree ($\bullet_T$) and co-tree ($\bullet_C$) branch voltage-current relationships, and the second and third rows consist of fundamental loop and fundamental cut-set equations for G_V and G_I, respectively.

Let the circuit have n nodes and b branches and contain k symbolic components (Y_{S1}, …, Y_{Sk}) in the co-tree branches (links) and l symbolic components (Z_{S1}, …, Z_{Sl}) in the tree branches; we define $w = b - n - k + 1$, $t = n - l - 1$. Diagonal matrices $\mathbf{Y}_C$ and $\mathbf{Z}_T$ can be partitioned as follows

$$\mathbf{Y}_C = \begin{bmatrix} \mathbf{Y}_{Cs} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{Cn} \end{bmatrix}, \quad \mathbf{Z}_T = \begin{bmatrix} \mathbf{Z}_{Ts} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_{Tn} \end{bmatrix} \tag{8.18}$$

where subscript s denotes immitances of symbolic components and subscript n denotes immitances of components given numerically.

Matrices $\mathbf{B}_T$ (fundamental loop matrix in G_V) and $\mathbf{Q}_C$ (fundamental cut-set matrix in G_I) can also be partitioned as follows:

$$\mathbf{B}_T = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}, \quad \mathbf{Q}_C = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix} \tag{8.19}$$

Rows of $\mathbf{B}_{11}$ and $\mathbf{B}_{12}$ correspond to symbolic co-tree branches (in G_V) and their columns correspond to symbolic and numeric tree branches, respectively. Rows of $\mathbf{B}_{21}$ and $\mathbf{B}_{22}$ correspond to numeric co-tree branches. Rows of $\mathbf{Q}_{11}$ and $\mathbf{Q}_{12}$ correspond to symbolic tree branches and their columns correspond to symbolic and numeric co-tree branches, respectively. Rows of $\mathbf{Q}_{21}$ and $\mathbf{Q}_{22}$ correspond to numeric tree branches. The submatrices are therefore of the following order: $\mathbf{B}_{11}$: $k \times l$, $\mathbf{B}_{22}$: $w \times t$, $\mathbf{Q}_{11}$: $l \times k$, $\mathbf{Q}_{22}$: $t \times w$.

Let $S_x = \{1, 2, \ldots, x\}$. For a given matrix **F** of order $a \times b$ let $\mathbf{F}(I_u, J_v)$ be the submatrix of **F** consisting of the rows and columns corresponding to the integers in the sets I_u, J_v, respectively. The sets $I_u = \{i_1, i_2, \ldots, i_u\}$ and $J_v = \{j_1, j_2, \ldots, j_v\}$ are subsets of S_a and S_b, respectively. Let us also introduce the following notation:

$$\mathbf{1}_d^c = \mathrm{diag}\begin{bmatrix} e_1 & e_2 & \cdots & e_d \end{bmatrix}; \quad c < d$$

$$e_x = \begin{cases} 0 \text{ for } x \in \{1, 2, \ldots, c\} \\ 1 \text{ for } x \in \{c+1, c+2, \ldots, d\} \end{cases}$$

The determinant of the system matrix **H** in Eq. (8.17), when some parameters take fixed numerical values, is

$$\begin{aligned} \det H = a &+ \sum_{J_v} b(\alpha_v) Z_{s_{j1}} Z_{s_{j2}} \cdots Z_{s_{jv}} + \sum_{I_u} c(\beta_u) Y_{s_{i1}} Y_{s_{i2}} \cdots Y_{s_{iu}} \\ &+ \sum_{I_u} \sum_{J_v} d(\alpha_v \beta_u) Z_{s_{j1}} Z_{s_{j2}} \cdots Z_{s_{jv}} Y_{s_{i1}} Y_{s_{i2}} \cdots Y_{s_{iu}} \end{aligned} \tag{8.20}$$

where the summations are taken over all possible symbol combinations α_v (symbolic tree elements) and β_u (symbolic co-tree elements), and the numerical coefficients are given by:

$$
\begin{aligned}
a &= \det\left[\mathbf{1}_w + \mathbf{B}'_{22}\left(-\mathbf{Q}'_{22}\right)\right] = \det\left[\mathbf{1}_t + \left(-\mathbf{Q}'_{22}\right)\mathbf{B}'_{22}\right] \\
b\left(\alpha_v\right) &= \det\left(\mathbf{1}^{v}_{t+v} + \begin{bmatrix} -\mathbf{Q}_{12}\left(J_v, I_w\right) \\ -\mathbf{Q}'_{22} \end{bmatrix}\begin{bmatrix} \mathbf{B}'_{21}\left(I_w, J_v\right) & \mathbf{B}'_{22} \end{bmatrix}\right) \\
c\left(\beta_u\right) &= \det\left(\mathbf{1}^{u}_{w+u} + \begin{bmatrix} \mathbf{B}_{12}\left(I_u, J_t\right) \\ \mathbf{B}'_{22} \end{bmatrix}\begin{bmatrix} -\mathbf{Q}'_{21}\left(J_t, I_u\right) & -\mathbf{Q}'_{22} \end{bmatrix}\right) \\
d\left(\alpha_v\beta_u\right) &= \det\left(\mathbf{1}^{v}_{v+t+u} + \begin{bmatrix} -\mathbf{Q}_{11}\left(J_v, I_u\right) & -\mathbf{Q}_{12}\left(J_v, I_w\right) \\ -\mathbf{Q}'_{21}\left(J_t, I_u\right) & -\mathbf{Q}'_{22} \\ \mathbf{1}_u & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{B}_{11}\left(I_u, J_v\right) & \mathbf{B}_{12}\left(I_u, J_t\right) & -\mathbf{1}_u \\ \mathbf{B}'_{21}\left(I_w, J_v\right) & \mathbf{B}'_{22} & \mathbf{0} \end{bmatrix}\right)
\end{aligned}
\tag{8.21}
$$

In the preceding equations, **0** represents a zero matrix of appropriate order, and the submatrices $\mathbf{B}'_{ij}$ and $\mathbf{Q}'_{ij}$ are defined as:

$$
\begin{aligned}
\mathbf{B}'_{21}\left(I_w, J_v\right) &= \mathbf{Y}_{Cn}\mathbf{B}_{21}\left(I_w, J_v\right), \quad \mathbf{B}'_{22} = \mathbf{Y}_{Cn}\mathbf{B}_{22} \\
\mathbf{Q}'_{21}\left(J_t, I_u\right) &= \mathbf{Z}_{Tn}\mathbf{Q}_{21}\left(J_t, I_u\right), \quad \mathbf{Q}'_{22} = \mathbf{Z}_{Tn}\mathbf{Q}_{22}
\end{aligned}
\tag{8.22}
$$

where the submatrix $\mathbf{B}_{21}(I_w, J_v)$ is obtained from the submatrix $\mathbf{B}_{21}$ by including all of its rows and only columns corresponding to a particular combination (α_v) of symbolic tree elements; submatrix $\mathbf{Q}_{21}(J_t, I_u)$ is obtained from the submatrix $\mathbf{Q}_{21}$ by including all of its rows and only columns corresponding to a particular combination (β_u) of symbolic co-tree elements.

Application of Eqs. (8.20) and (8.21) for a circuit with m symbolic parameters requires, theoretically, the calculation of 2^m determinants. Not all of these determinants may need to be calculated due to the following property of the determinants in Eq. (8.21). If a set of symbolic tree elements (α_v) forms a cut-set in G_I (*symbolic tree cut-set*), then the corresponding coefficients $b(\alpha_v)$ and $d(\alpha_v\beta_u)$ in Eq. (8.20) equal to zero. Likewise, if the set of symbolic co-tree elements (β_u) forms a loop in G_V (*symbolic co-tree loop*), the corresponding coefficients $c(\beta_u)$ and $d(\alpha_v\beta_u)$ in Eq. (8.20) equal to zero.

Once the determinant det(**H**) is obtained from Eq. (8.20), the sorting scheme, identical to that expressed in Eq. (8.9), is applied and the required network function(s) can be calculated using Eqs. (8.6) through (8.8).

The main feature of this approach is the fact that each coefficient at a valid symbol combination is obtained directly by calculating a single, easily formulated determinant (a polynomial in s, in general case). The method was implemented in a computer program called UTSSNAP [52]. The following example illustrates this technique of parameter extraction.

Example 5. Consider again the circuit in Figure 8.8. Assume this time that two components, R_1 and g_m, are given symbolically. The goal is again to find the input impedance Z_{41} in a semi-symbolic form using the parameter extraction method based on the two-graph tableau formulation.

The voltage and current graphs of the circuit are shown in Figure 8.10. The common spanning tree chosen is $T = \{R_1, R_2, R_3\}$ with one symbolic element. For this circuit, we have: $n = 4$, $b = 7$, $k = 2$, $l = 1$, $w = 2$, and $t = 2$.

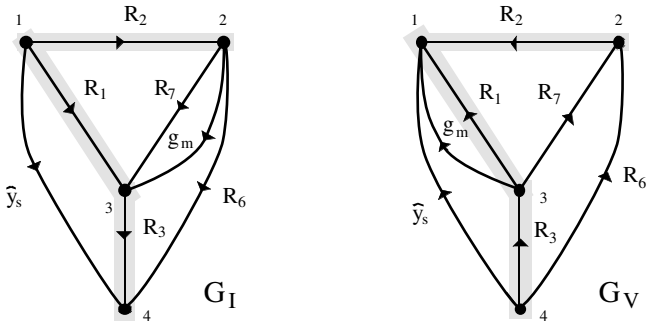

FIGURE 8.10 The current and voltage graphs for the circuit in Figure 8.8 with the common spanning tree highlighted.

The matrices $\mathbf{Y}_C$, $\mathbf{Z}_T$, $\mathbf{Q}_C$, and $\mathbf{B}_T$ can now be determined as:

$$\mathbf{Y}_C = \left[\begin{array}{cc|c} \hat{y}_S & & \\ & g_m & \\ \hline & & 0.1 \\ & & & 10 \end{array}\right] \qquad \mathbf{Z}_T = \left[\begin{array}{cc|c} R_1 & & \\ \hline & & 0.2 \\ & & & 0.5 \end{array}\right]$$

$$\mathbf{Q}_C = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ \hline 0 & -1 & -1 & -1 \\ 1 & 0 & 1 & 0 \end{array}\right] \qquad \mathbf{B}_T = \left[\begin{array}{c|cc} -1 & 0 & -1 \\ -1 & 0 & 0 \\ \hline -1 & 1 & -1 \\ -1 & 1 & 0 \end{array}\right]$$

Using Eq. (8.22), we can calculate matrices $\mathbf{B}'_{22}$ and $\mathbf{Q}'_{22}$:

$$\mathbf{B}'_{22} = \begin{bmatrix} 0.1 & 0 \\ 0 & 10 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.1 & -0.1 \\ 10 & 0 \end{bmatrix}, \quad \mathbf{Q}'_{22} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -0.2 & -0.2 \\ 0.5 & 0 \end{bmatrix}$$

Now, applying Eq. (8.21), the coefficient a in Eq. (8.20) is calculated as:

$$a = \det\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.1 & -0.1 \\ 10 & 0 \end{bmatrix}\begin{bmatrix} 0.2 & 0.2 \\ -0.5 & 0 \end{bmatrix}\right) = \det\begin{bmatrix} 1.07 & 0.02 \\ 2 & 3 \end{bmatrix} = 3.17$$

Because only one symbolic tree element exists, namely R_1, we have: $\alpha_v = \{R_1\}$ and the associated sets: $J_v = \{1\}$, $I_w = \{1,2\}$. Using Eq. (8.22), we calculate

$$\mathbf{B}'_{21}(I_w, J_v) = \mathbf{Y}_{Cn}\mathbf{B}_{21}(I_w, J_v) = \begin{bmatrix} 0.1 & 0 \\ 0 & 10 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.1 \\ -10 \end{bmatrix}$$

The coefficient $b(R_1)$ can now be obtained from:

$$b(R_1) = \det\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \left[\begin{array}{cc} -1 & -1 \\ \hline 0.2 & 0.2 \\ -0.5 & 0 \end{array}\right]\left[\begin{array}{c|cc} -0.1 & 0.1 & -0.1 \\ -10 & 10 & 0 \end{array}\right]\right)$$

$$= \det\begin{bmatrix} 10.1 & -10.1 & 0.1 \\ -2.02 & 3.02 & -0.02 \\ 0.05 & -0.05 & 1.05 \end{bmatrix} = 10.6$$

Other numerical coefficients in Eq. (8.20) are calculated in a similar way:

$$c(\hat{y}_S) = 1.51,\ c(g_m) = 0,\ c(\hat{y}_S g_m) = 0,\ d(R_1\hat{y}_S) = 8.12,\ d(R_1 g_m) = 1.05,\ d(R_1\hat{y}_S g_m) = 0.51$$

Adding all terms, sorting according to Eq. (8.9), and applying Eq. (8.6) finally results in:

$$Z_{41} = \frac{1.51 + 8.12R_1 + 0.51R_1 g_m}{3.17 + 10.6R_1 + 1.05R_1 g_m}$$

Matrices in Eq. (8.21) may contain terms dependent on the complex frequency s. Determinants of such matrices are polynomials in s as long as all matrix elements are of the form: $a = \alpha + s\beta$. An interpolation method may be used to calculate the coefficients of those polynomials. One such method is briefly described in the next paragraph.

4. The interpolation method

This method is best suited when s is the only symbolic variable. In such case, a transfer function has the rational form

$$H(s) = \frac{N(s)}{D(s)} = \frac{\sum_{i=0}^{m} a_i s^i}{\sum_{i=0}^{n} b_i s^i}$$

where $N(s)$ and $D(s)$ are polynomials in s with real coefficients and $m \le n$.

Coefficients of an nth-order polynomial

$$P(s) = \sum_{k=0}^{n} p_k s^k$$

can be obtained by calculating the value of $P(s)$ at $n + 1$ distinct points s_i and then solving the following set of equations:

$$\begin{bmatrix} 1 & s_0 & s_0^2 & \cdots & s_0^n \\ 1 & s_1 & s_1^2 & \cdots & s_1^n \\ & \vdots & & & \\ 1 & s_n & s_n^2 & \cdots & s_n^n \end{bmatrix}\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} P(s_0) \\ P(s_1) \\ \vdots \\ P(s_n) \end{bmatrix} \tag{8.23}$$

Because the matrix in Eq. (8.23) is nonsingular, the unique solution exists. It is well known [58, 64] that for numerical accuracy and stability, the best choice of the interpolation points is a set of $q \geq n + 1$ points s_i uniformly spaced on the unit circle in the complex plane. Once all the values of $P(s_i)$ are known, the polynomial coefficients can be calculated through the discrete Fourier transform (DFT).

To apply this technique to the problem of finding a transfer function, let us assume that a circuit behavior is described by a linear equation

$$\mathbf{Ax} = \mathbf{b} \tag{8.24}$$

in which the coefficient matrix has entries of the form: $a = \alpha + s\beta$ (both the modified nodal and the tableau methods have this property). Then, each transfer function of such circuit has the same denominator $D(s) = |\mathbf{A}|$. If the circuit Eq. (8.24) is solved by LU factorization at $s = s_i$, both the transfer function $H(s_i)$ and its denominator $D(s_i)$ are obtained simultaneously. The value of the numerator is then calculated simply as $N(s_i) = H(s_i)D(s_i)$. Repeating this process for all points s_i ($i = 0, 1, \ldots, q$) and then applying the DFT to both sets of values, $D(s_i)$ and $N(s_i)$, gives the required coefficients of the numerator and denominator polynomials.

If the number of interpolation points is an integer power of 2 ($q = 2^k$), the method has the advantage that the fast Fourier transform can be used to find the coefficients. This greatly enhances the execution time [40]. The method has been extended to handle several symbolic variables in addition to s [58]. The program implementation [64] allows a maximum of five symbolic parameters in a circuit.

With the emergence of approximate symbolic analysis, the polynomial interpolation method has attracted new interest. (It is desirable to know the accurate numerical value of polynomial coefficients before one attempts an approximation.) Recently, a new adaptive scaling mechanism was proposed [14] that significantly increases the circuit size that can be handled accurately and efficiently.

Other classifications of symbolic methods have been reported [18]. These methods can be considered as variations on the previous basic four methods. The reported methods include elimination algorithms, recursive determinant-expansion algorithms, and nonrecursive nested-minors method. All three are based on the use of Cramer's rule to find the determinant and the cofactors of a matrix. Another reported class of algorithms uses Modified Nodal Analysis [27] as the basis of the analysis, sometimes referred to as a direct network approach [22, 36]. This class of methods is covered in the next section.

The first generation of computer programs available for symbolic circuit simulation based on these methods includes NASAP [47] and SNAP [38]. Research in the late 1980s and early 1990s produced newer symbolic analysis programs. These programs include ISSAC [18], SCAPP [22], ASAP [12], EASY [60], SYNAP [57], SAPEC [43], SAPWIN [37], SCYMBAL [31], GASCAP [29], SSPICE [66], and STAINS [53].

8.4 Hierarchical Methods (Sequence of Expressions)

All the methods presented in the previous section have circuit size limitations. The main problem is the exponential growth of the number of symbolic terms involved in the expression for the transfer function in Eq. (8.3) as the circuit gets larger. The solution to analyzing large-scale circuits lies in a total departure from the traditional procedure of trying to state the transfer function as a single expression and using a *sequence of expressions* (SoE) procedure instead. The idea is to produce a succession of small expressions with a backward hierarchical dependency on each other. The growth of the number of expressions in this case will be, at worst case, quadratic [22].

The advantage of having the transfer function stated in a single expression lies in the ability to gain insight to the relationship between the transfer function and the network elements by inspection [39]. For large expressions, though, this is not possible and the single expression loses that advantage. ISSAC [67], ASAP [13], SYNAP [57], and Analog Insydes [26] attempt to handle larger circuits by maintaining the single expression method and using circuit dependent approximation techniques. The tradeoff is

accuracy for insight. Therefore, the SoE approach is more suitable for accurately handling large-scale circuits. The following example illustrates the features of the sequence of expressions.

Example 6. Consider the resistance ladder network in Figure 8.11. The goal is to obtain the input impedance function of the network, $Z_{in} = V_{in}/I_{in}$. The single expression transfer function Z_4 is:

$$Z_4 = \frac{R_1R_3 + R_1R_4 + R_2R_3 + R_2R_4 + R_3R_4}{R_1 + R_2 + R_3}$$

The number of terms in the numerator and denominator are given by the Fibonacci numbers satisfying the following difference equation:

$$y_{k+2} = y_{k+1} + y_k; \quad k = 0,1,2,\ldots; \quad y_0 = 0,\ y_1 = 1$$

An explicit solution to the preceding equation is:

$$y_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \approx 0.168 \cdot 1.618^n \text{ for large } n$$

The solution demonstrates that the number of terms in Z_n increases exponentially with n. Any single expression transfer function has this inherent limitation.

Now, using the SoE procedure, the input impedance can be obtained from the following expressions:

$$Z_1 = R_1; \quad Z_2 = Z_1 + R_2; \quad Z_3 = \frac{Z_2R_3}{Z_2 + R_3}; \quad Z_4 = Z_3 + R_4$$

It is obvious for each additional resistance added, the sequence of expressions will grow by one expression, either of the form $Z_{i-1} + R_i$ or $Z_{i-1}R_i/Z_{i-1} + R_i$. The number of terms in the sequence of expressions can be calculated from the formula:

$$y_n = \begin{cases} 2.5n - 2 & \text{for } n \text{ even} \\ 2.5n - 1.5 & \text{for } n \text{ odd} \end{cases}$$

which exhibits a linear growth with respect to n. Therefore, to find the input impedance of a 100-resistor ladder network, the single expression methods would produce 7.9×10^{20} terms, which requires unrealistically huge computer storage capabilities. On the other hand, the SoE method would produce only 248 terms, which is even within the scope of some desk calculators.

Another advantage of the SoE is the number of arithmetic operations needed to evaluate the transfer function. To evaluate Z_9, for example, the single expression methods would require 302 multiplications

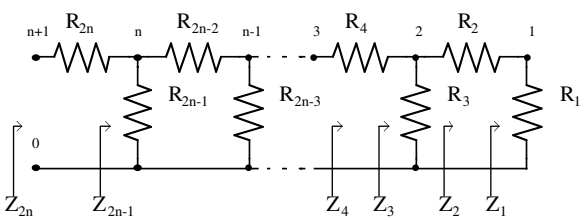

FIGURE 8.11 Resistive ladder network.

and 87 additions. The SoE method would only require eight multiplications and eight additions, a large reduction in computer evaluation time. All this makes the concept of symbolic circuit simulation of large-scale networks very possible.

Two topological analysis methods for symbolic simulation of large-scale circuits have been proposed in [61] and in [25]. The first method utilizes the SoE idea to obtain the transfer functions. The method operates on the Coates graph [8] representing the circuit. A partitioning is proposed onto the flowgraph and not the physical network. The second method also utilizes the sequence of expressions and a Mason's signal flow graph [45] representation of the circuit. The method makes use of partitioning on the physical level instead of on the graph level. Therefore, for a hierarchical circuit, the method can operate on the subcircuits in a hierarchical fashion in order to produce a final solution. The fundamentals of both signal flow graph methods were described in the previous section.

Another hierarchical approach is one that is based on Modified Nodal Analysis [27]. This method [22] exhibits a linear growth (for practical circuits) in the number of terms in the symbolic solutions. The analysis methodology introduces the concept of the RMNA (Reduced Modified Nodal Analysis) matrix. This allows the characterization of symbolic circuits in terms of only a small subset of the network variables (external variables) instead of the complete set of variables. The method was made even more effective by introducing a locally optimal pivot selection scheme during the reduction process [53]. For a circuit containing several identical[4] subcircuits, the analysis algorithm is most efficient when network partitioning is used. For other circuits, the best results (the most compact SoE) are obtained when the entire circuit is analyzed without partitioning.

The SoE generation process starts with the formulation of a symbolic Modified Node Admittance Matrix (MNAM) for a circuit [40, 64]. Then all internal variables are suppressed one by one using Gaussian elimination with locally optimal pivot selection. Each elimination step produces a series of expressions and modifies some entries in the remaining portion of the MNAM. When all internal variables are suppressed, the resulting matrix is known as the Reduced Modified Node Admittance Matrix (RMNAM). Usually it will be a 2×2 matrix of a two-port.[5] Most transfer functions of interest to a circuit designer can be represented by formulas involving the elements of RMNAM and the terminating admittances. A detailed discussion of the method can be found in [53]. Based on this approach, a computer program called STAINS was developed.

For a circuit with several identical subcircuits, the reduction process is first applied to all internal variables[6] of the subcircuit, resulting in an intermediate RMNAM describing the subcircuit. Those RMNAMs are then recombined with the MNAM of the remaining circuit and the reduction process is repeated on the resulting matrix.

To further illustrate the SoE approach, we present the following example.

Example 7. Consider a bipolar cascade stage with bootstrap capacitor C_B illustrated in Figure 8.12 [18]. With the BJTs replaced by their low-frequency hybrid-π models (with r_B, g_m, and r_o only), the full symbolic analysis yields the output admittance formula outlined in Figure 8.13. The formula requires 48 additions and 117 multiplication/division operations. STAINS can generate several different sequences of expressions. One of them is presented in Figure 8.14. It requires only 24 additions and 17 multiplications/divisions.[7]

[4]The subcircuits have to be truly identical, i.e., they must have the same topology and component symbols. A typical example would be a large active filter containing a number of identical, nonideal op amps.

[5]In sensitivity calculations using SoE [5], the final RMNAM may need to be larger than 2×2.

[6]The internal variables are the variables not directly associated with the subcircuit's connections to the rest of the circuit.

[7]Counting of visible arithmetic operations gives only a rough estimate of the SoE complexity, especially when complex numbers are involved. Issues related to SoE computational efficiency are discussed in Reference [55].

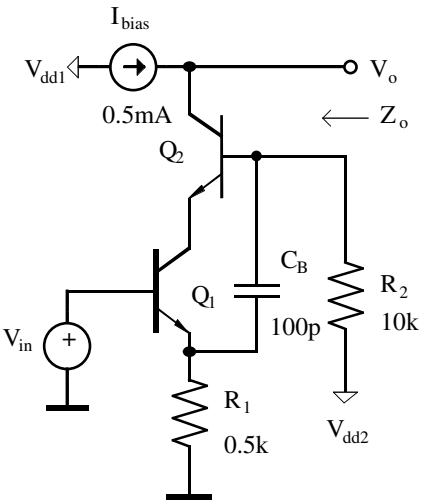

FIGURE 8.12 Bipolar cascode stage.

```
Zo = (G2*Gm1*Gm2+G1*G2*Gm1+G2*Gm2*Gp1+G2*Gm1*Gp2+Gm2*Go1*Gp1+G1*G2*Gp1+G2*Gm2*Go1+...
    G2*Gm1*Go2+G1*Go2*Gp1+G2*Gp1*Gp2+G1*Go1*Gp1+G1*G2*Go2+G1*G2*Go1+Go2*Gp1*Gp2+...
    Go1*Gp1*Gp2+G2*Go2*Gp1+G2*Go1*Gp2+G2*Go2*Gp2+Go1*Go2*Gp1+G2*Go1*Go2+...
    s*(Cb*Gm1*Gm2+Cb*G1*Gm1+Cb*Gm1*Gp2+Cb*G2*Gm1+Cb*G1*Gp1+Cb*Gm2*Go1+Cb*Gp1*Gp2+...
    Cb*G1*Go2+Cb*G2*Gp1+Cb*G1*Go1+Cb*Go1*Gp2+Cb*Go2*Gp2+Cb*Go1*Gp1+Cb*G2*Go1+...
    Cb*G2*Go2+Cb*Go1*Go2))/...
    (Go1*G2*Gm2*Gp1+Go1*G1*G2*Gp1+Go1*G2*Gp1*Gp2+Go1*G1*Go2*Gp1+Go1*G1*G2*Go2+...
    Go1*Go2*Gp1*Gp2+Go1*G2*Go2*Gp2+Go1*G2*Gp2*Gp1+...
    s*(Cb*Go1*G1*Gp1+Cb*Go1*Gp1*Gp2+Cb*Go1*G2*Gp1+Cb*Go1*G1*Go2+Cb*Go1*Go2*Gp2+...
    Cb*Go1*G2*Go2));
```

FIGURE 8.13 Full symbolic expression for Z_o of the cascode in Figure 8.12.

```
d1 = -(G2+Gp2+s*Cb)/(s*Cb);
x1 = (Go1+Gm1)*d1-Gp2-Gm2;
x2 = -s*Cb-(G1+Gp1+Go1+Gm1+s*Cb)*d1;
d2 = Gp2/(s*Cb);
x3 = Go1+Gp2+Go2+Gm2+(Go1+Gm1)*d2;
x4 = -Go1-(G1+Gp1+Go1+Gm1+s*Cb)*d2;
d3 = x2/(x4);
x5 = Gm2+(Go2+Gm2)*d3;
x6 = x1-x3*d3;
Yo = Go2+x5*Go2/(x6);
Zo = 1/Yo;
```

FIGURE 8.14 The SoE generated by STAINS for the cascode in Figure 8.12.

8.5 Approximate Symbolic Analysis

The SoE approach offers a solution for the exact symbolic analysis of large circuits. For some applications, it may be more important to obtain a simpler inexact expression, but the one that would clearly identify the dominant circuit components and their role in determining circuit behavior. Approximate symbolic

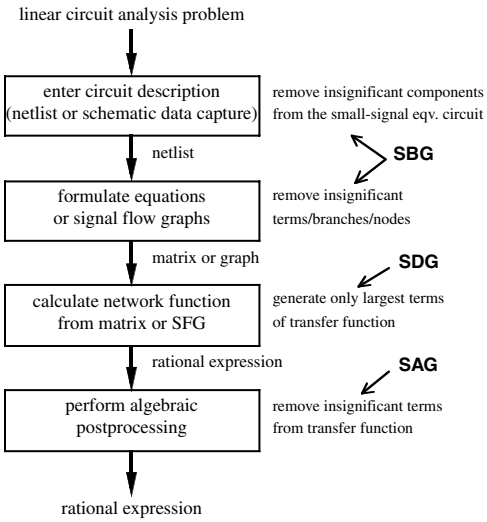

FIGURE 8.15 Classification of symbolic approximation techniques [26].

analysis provides the answer. Of course, manual approximation (simplification) techniques have been known and practiced by engineers for decades. To obtain compact and meaningful expressions by computer, symbolic analysis software must be capable of performing those approximations that are applied in manual circuit analysis in an automatic fashion. In addition to that, computer algorithms should be able to employ simplification strategies not available (or impractical) in manual approximation.

In the last decade, a number of symbolic approximation algorithms have been developed and implemented in symbolic circuit analysis programs. Depending on the stage in the circuit analysis process in which they are applied, these algorithms can be categorized as: *simplification before generation* (SBG), *simplification during generation* (SDG), and *simplification after generation* (SAG). Figure 8.15, adapted from [26], presents an overview of the three types of approximation algorithms.

SBG involves removing circuit components and/or individual entries in the circuit matrix (the *sifting* approach [28]) or eliminating some graph branches (the *sensitivity-based two-graph simplification* [69]) that do not contribute significantly to the final formula.

SDG is based on generation of symbolic terms in a decreasing order of magnitude. The generation process is stopped when the error reaches the specified level. The most successful approach to date is based on the two-graph formulation [68]. It employs an algorithm to generate the common spanning trees in strictly decreasing order of magnitude [30]. In the case of frequency-dependent circuit, this procedure is applied separately to different powers of s. Mathematical formalism of *matroids* is well suited to describe problems of SDG [69].

When applied alone, SAG is a very ineffective technique, because it requires generation and storage of a large number of unnecessary terms. When combined with SBG and SDG methods, however, it can produce the most compact expressions by pruning redundant terms not detected earlier in the simplification process.

All simplification techniques require careful monitoring of the approximation amplitude and phase errors (ε_A and ε_p). The error criteria can be expressed as follows:

$$\left|\frac{\left|H(s,\mathbf{x})\right|-\left|H^*(s,\mathbf{x})\right|}{\left|H(s,\mathbf{x})\right|}\right| \le \varepsilon_A$$

$$\left|\angle H(s,\mathbf{x})-\angle H^*(s,\mathbf{x})\right| \le \varepsilon_p$$

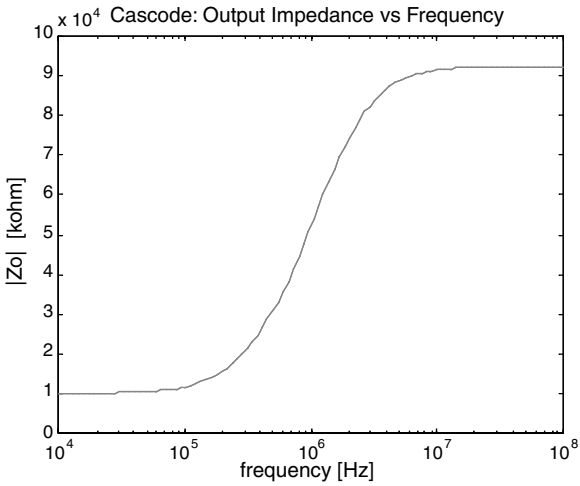

FIGURE 8.16 Plot of $|Z_o|$ of the cascode, obtained numerically from the exact formula.

for $s = j\omega$, $\omega \in (\omega_1,\omega_2)$, and $\mathbf{x} \in (\mathbf{x}_1,\mathbf{x}_2)$, where $H(s,\mathbf{x})$ is the exact transfer function, defined by Eq. (8.3), and $H^*(s,\mathbf{x})$ is the approximating function. The majority of the approximation methods developed to date use the simplified criteria, where the errors are measured only for a given set of circuit parameters $\mathbf{x}^0$ (the nominal design point) [33].

The following example, although quite simple, illustrates very well the advantages of approximate symbolic analysis.

Example 8 [18]. Consider again the bipolar cascode stage, depicted in Figure 8.12 and its fully symbolic expression for the output impedance, depicted in Figure 8.13. Even for such a simple circuit, the full symbolic result is very hard to interpret and therefore not able to provide insight into the circuit behavior. Sequential form of the output impedance formula, presented in Figure 8.14, is more compact than the full expression but also cannot be utilized for interpretation.

A plot of $|Z_o|$ for a nominal set of component values (r_π = 5 kΩ, g_m = 20 mS, r_o = 100 kΩ for both BJTs), obtained numerically from the SoE in Figure 8.14, is plotted in Figure 8.16. By examining the plot, one can appreciate the general behavior of the function, but it is difficult to predict the influence of various circuit components on the output impedance.

Applying symbolic approximation techniques we can obtain less accurate but still more revealing formulas. If a 10% maximum amplitude error is accepted, the simplified function takes the following form:

$$Z_{o(10\%)} = \frac{g_{m1}\left(g_{m2}+G_1\right)\left(G_2+sC_B\right)}{g_{o1}g_{\pi1}\left[G_2\left(g_{m2}+G_1\right)+sC_B\left(G_1+g_{\pi2}\right)\right]}$$

If we allow a 25% magnitude error,[8] the output impedance formula can be simplified further:

$$Z_{o(25\%)} = \frac{g_{m1}g_{m2}\left(G_2+sC_B\right)}{g_{o1}g_{\pi1}\left(G_2g_{m2}+sC_BG_1\right)} \tag{8.25}$$

[8]It is important to note that the approximate expressions were developed taking into account variations of BJT parameters; the fact that both simplified formulas give identical results at the nominal design point is purely coincidental.

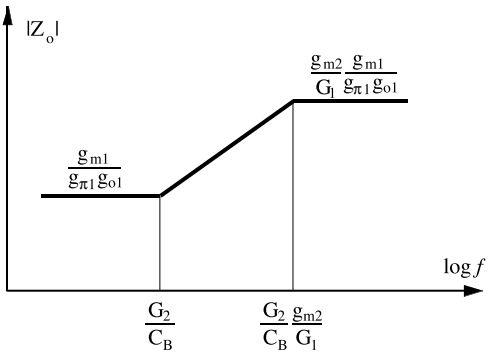

FIGURE 8.17 Asymptotic plot of $|Z_o|$ of the cascode based on Eq. (8.26).

The impedance levels as well as pole and zero estimates can be easily obtained from Eq. (8.25):

$$
\begin{aligned}
Z_o(\text{low } f) &\cong \frac{g_{m1}}{g_{\pi 1} g_{o1}} = \frac{\beta_1}{g_{o1}} \\
Z_o(\text{high } f) &\cong \frac{g_{m1} g_{m2}}{g_{\pi 1} g_{o1} G_1} = \frac{g_{m2}}{G_1} Z_o(\text{low } f) \\
z &\cong -\frac{G_2}{C_B} \\
p &\cong -\frac{g_{m2} G_2}{G_1 C_B}
\end{aligned}
\tag{8.26}
$$

An asymptotic plot of $|Z_o|$, based on Eq. (8.26), is plotted in Figure 8.16.

8.6 Time-Domain Analysis

The previous sections discussed the different frequency domain techniques for symbolic analysis. Symbolic analysis methods in the transient domain did not appear until the beginning of the 1990s [3, 24, 36]. The main limitation to symbolic time-domain analysis is the difficulty in handling the symbolic integration and differentiation needed to handle the energy storage elements (mainly capacitors and inductors). This problem, of course, does not exist in the frequency domain because of the use of Laplace transforms to represent these elements. Although symbolic algebra software packages are available, such as MATHMATICA, MAXIMA, and MAPLE, which can be used to perform integration and differentiations, they have not been applied to transient symbolic analysis due to the execution time complexity of these programs. All but one of the approaches in the time domain are actually semi-symbolic. The semi-symbolic algorithms use a mixture of symbolic and numeric techniques to perform the analysis. The work here is still in its infancy. This section briefly discusses the three contributions published in the literature thus far.

All symbolic time domain techniques deal with linear circuits and can be classified under one of the two categories.

Fully Symbolic

Only one method has been reported in the literature that is fully symbolic [20]. This method utilizes a direct and hierarchical symbolic transient analysis approach similar to the one reported in [22]. The formulation is based on the well-known discrete models for numerical integration of linear differential

equations. Three of these integration methods are implemented symbolically: the Backward Euler, the Trapezoidal, and Gear's 2nd-Order Backward Differentiation [20]. The inherent accuracy problems due to the approximations in these methods show up when the symbolic expressions are evaluated numerically. A detailed discussion of this method can be found in [20].

Semi-Symbolic

Three such algorithms have been reported in the literature thus far. Two of them [24, 36] simply take the symbolic expressions in the frequency domain, evaluate them numerically for a range of frequencies, and then perform a numeric inverse laplace transformation or a fast Fourier transformation (FFT) on the results. The approach reported in [36] uses an MNA, then a state-variable symbolic formulation to get the frequency domain response and can handle time-varying circuits, namely, switch power converters. The approach in [24] uses a hierarchical network approach [22] to generate the symbolic frequency domain response. The third algorithm reported in [3] is a hierarchical approach that uses an MNA and a state-variable symbolic formulation and then uses the eigenvalues of the system to find a closed-form numerical transient solution.

References

[1] G. E. Alderson, P. M. Lin, "Integrating Topological and Numerical Methods for Semi-Symbolic Network Analysis," *Proc. of the 13th Midwest Symposium on Circuit Theory*, 1970.

[2] G. E. Alderson, P. M. Lin, "Computer Generation of Symbolic Network Functions — A New Theory and Implementation," *IEEE Trans. on Circuit Theory*, vol. CT-20, pp. 48–56, Jan. 1973.

[3] B. Alspaugh, M. Hassoun, "A Mixed Symbolic and Numeric Method for Closed-Form Transient Analysis," *Proc. ECCTD*, Davos, 1993.

[4] Z. Arnautovic, P. M. Lin, "Symbolic Analysis of Mixed Continuous and Sampled Data Systems," *Proc. IEEE ISCAS*, pp. 798–801, 1991.

[5] F. Balik, B. Rodanski, "Calculation of First-Order Symbolic Sensitivities in Sequential Form via the Transimpedance Method," *Proc. SMACD*, Kaiserslautern, Germany, Oct. 1998, pp. 169–172.

[6] D. A. Calahan, "Linear Network Analysis and Realization — Digital Computer Programs and Instruction Manual," University of Ill. Bull., vol. 62, Feb. 1965.

[7] L. O. Chua, P. M. Lin, *Computer-Aided Analysis of Electronic Circuits — Algorithms and Computational Techniques*. Englewood Cliffs, NJ: Prentice Hall, 1975.

[8] C. L. Coates, "Flow graph Solutions of Linear Algebraic Equations," *IRE Trans. on Circuit Theory*, vol. CT-6, pp. 170–187, 1959.

[9] F. Constantinescu, M. Nitescu, "Computation of Symbolic Pole/Zero Expressions for Analog Circuit Design," *Proc. SMACD*, Haverlee, Belgium, Oct. 1996.

[10] G. DiDomenico et al., "BRAINS: A Symbolic Solver for Electronic Circuits," *Proc. SMACD*, Paris, Oct. 1991.

[11] G. Dröge, E. H. Horneber, "Symbolic Calculation of Poles and Zeros," *Proc. SMACD*, Haverlee, Belgium, Oct. 1996.

[12] F. V. Fernandez, A. Rodriguez-Vazquez, J. L. Huertas, "An Advanced Symbolic Analyzer for the Automatic Generation of Analog Circuit Design Equations," *Proc. IEEE ISCAS*, Singapore, pp. 810–813, June 1991.

[13] F. V. Fernandez *et al.*, "On Simplification Techniques for Symbolic Analysis of Analog Integrated Circuits," *Proc. IEEE ISCAS*, San Diego, CA, pp. 1149–1152, May 1992.

[14] F. V. Fernandez *et al.*, "Symbolic Analysis of Large Analog Integrated Circuits: The Numerical Reference Generation Problem," *IEEE Trans. on Circuits and Systems — II: Analog and Digital Signal Processing*, vol. 45, no. 10, pp. 1351–1361, Oct. 1998.

[15] J. K. Fidler, J. I. Sewell, "Symbolic Analysis for Computer-Aided Circuit Design — The Interpolative Approach," *IEEE Trans. on Circuit Theory*, vol. CT-20, Nov. 1973.

[16] T. F. Gatts, N. R. Malik, "Topoloigical Analysis Program For Linear Active Networks (TAPLAN)," *Proc. of the 13th Midwest Symposium on Circuit Theory*, 1970.
[17] G. Gielen, H. Walscharts, W. Sansen, "ISSAC: A Symbolic Simulator for Analog Integrated Circuits," *IEEE J. of Solid-State Circuits*, vol. SC-24, pp. 1587–1597, Dec. 1989.
[18] G. Gielen, W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*. Boston, MA: Kluwer Academic, 1991.
[19] S. Greenfield, *Transient Analysis for Symbolic Simulation*, MS Thesis, Iowa State University, Dec. 1993.
[20] S. Greenfield, M. Hassoun, "Direct Hierarchical Symbolic Transient Analysis of Linear Circuits," *Proc. ISCAS*, 1994.
[21] G. D. Hachtel et al., "The Sparse Tableau Approach to Network and Design," *IEEE Trans. on Circuit Theory*, vol. CT-18, pp. 101–113, Jan 1971.
[22] M. M. Hassoun, P. M. Lin, "A New Network Approach to Symbolic Simulation of Large-Scale Networks," *Proc. IEEE ISCAS*, pp. 806–809, May 1989.
[23] M. M. Hassoun, P. M. Lin, "An Efficient Partitioning Algorithm for Large-Scale Circuits," Proc. *IEEE ISCAS*, New Orleans, pp. 2405–2408, May 1990.
[24] M. M. Hassoun, J. E. Ackerman, "Symbolic Simulation of Large Scale Circuits in Both Frequency and Time Domains," *Proc. IEEE MWSCAS*, Calgary, pp. 707–710, Aug. 1990.
[25] M. Hassoun, K. McCarville, "Symbolic Analysis of Large-Scale Networks Using a Hierarchical Signal Flow Graph Approach," *J. of Analog VLSI and Signal Processing*, Jan. 1993.
[26] E. Henning, Symbolic Approximation and Modeling Techniques for Analysis and Design of Analog Circuits. Doctoral Dissertation, University of Kaiserslautern. Aachen: Shaker Verlag, 2000.
[27] C. Ho, A. E. Ruehli, P. A. Brennan, "The Modified Nodal Approach to Network Analysis," *IEEE Trans. on Circuits and Systems*, vol. CAS-25, pp. 504–509, June 1975.
[28] J. J. Hsu, C. Sechen, "Low-Frequency Symbolic Analysis of Large Analog Integrated Circuits," *Proc. CICC*, 1993, pp. 14.7.1–14.7.4.
[29] L. Huelsman, "Personal Computer Symbolic Analysis Programs for Undergraduate Engineering Courses," *Proc. ISCAS*, pp. 798–801, 1989.
[30] N. Katoh, T. Ibaraki, H. Mine, "An Algorithm for Finding k Minimum Spanning Trees," *SIAM J. Comput.*, vol. 10, no. 2, pp. 247–255, May 1981.
[31] A. Konczykowska, M. Bon, "Automated Design Software for Switched Capacitor ICs with Symbolic Simulator SCYMBAL," *Proc. DAC*, pp. 363–368, 1988.
[32] A. Konczykowska et al., "Symbolic Analysis as a Tool for Circuit Optimization," *Proc. IEEE ISCAS*, San Diego, CA, pp. 1161–1164, May 1992.
[33] A. Konczykowska, "Symbolic circuit analysis," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. New York: John Wiley & Sons, 1999.
[34] J. Lee, R. Rohrer, "AWEsymbolic: Compiled Analysis of Linear(ized) Circuits Using Asymptotic Waveform Evaluation," *Proc. DAC*, pp. 213-218, 1992.
[35] B. Li, D. Gu, "SSCNAP: A Program for Symbolic Analysis of Switched Capacitor Circuits," *IEEE Trans. on CAD*, vol. 11, pp. 334–340, 1992.
[36] A. Liberatore et al., "Simulation of Switching Power Converters Using Symbolic Techniques," *Alt Frequenza*, vol. 5, no. 6, Nov. 1993.
[37] A. Liberatore et al., "A New Symbolic Program Package for the Interactive Design of Analog Circuits," *Proc. IEEE ISCAS*, Seattle, WA, pp. 2209–2212, May 1995.
[38] P. M. Lin, G. E. Alderson, "SNAP — A Computer Program for Generating Symbolic Network Functions," School of EE, Purdue University, West Lafayette, IN, Rep. TR-EE 70-16, Aug. 1970.
[39] P. M. Lin, "A Survey of Applications of Symbolic Network Functions," *IEEE Trans. on Circuit Theory*, vol. CT-20, pp. 732–737, Nov. 1973.
[40] P. M. Lin, *Symbolic Network Analysis*. Amsterdam: Elsevier Science, 1991.
[41] P. M. Lin, "Sensitivity Analysis of Large Linear Networks Using Symbolic Programs," *Proc. IEEE ISCAS*, San Diego, CA, pp. 1145–1148, May 1992.

[42] V. K. Manaktala, G. L. Kelly, "On the Symbolic Analysis of Electrical Networks," *Proc. of the 15th Midwest Symposium on Circuit Theory*, 1972.
[43] S. Manetti, "New Approaches to Automatic Symbolic Analysis of Electric Circuits," *Proc. IEE*, pp. 22–28, Feb. 1991.
[44] M. Martins et al., "A Computer-Assisted Tool for the Analysis of Multirate SC Networks by Symbolic Signal Flow Graphs," *Alt Frequenza*, vol. 5, no. 6, Nov. 1993.
[45] S. J. Mason, "Feedback Theory — Further Properties of Signal Flow Graphs," *Proc. IRE*, vol. 44, pp. 920–926, July 1956.
[46] J. O. McClanahan, S. P. Chan, "Computer Analysis of General Linear Networks Using Digraphs," *Int. J. of Electronics*, no. 22, pp. 153–191, 1972.
[47] L.P. McNamee, H. Potash, *A User's and Programmer's Manual for NASAP*, University of California at Los Angeles, Rep. 63-38, Aug. 1968.
[48] R. R. Mielke, "A New Signal Flowgraph Formulation of Symbolic Network Functions," *IEEE Trans. on Circuits and Systems*, vol. CAS-25, pp. 334–340, June 1978.
[49] H. Okrent, L. P. McNamee, *NASAP-70 User's and Programmer's Manual*, UCLA, Technical Report ENG-7044, 1970.
[50] M. Pierzchala, B. Rodanski, "A New Method of Semi-Symbolic Network Analysis," *Proc. IEEE ISCAS*, Chicago, IL, pp. 2240–2243, May 1993.
[51] M. Pierzchala, B. Rodanski, "Efficient Generation of Symbolic Network Functions for Large-Scale Circuits," *Proc. MWSCAS*, Ames, IO, pp. 425–428, August 1996.
[52] M. Pierzchala, B. Rodanski, "Direct Calculation of Numerical Coefficients in Semi-Symbolic Circuit Analysis," *Proc. SMACD*, Kaiserslautern, Germany, Oct. 1998, pp. 173–176.
[53] M. Pierzchala, B. Rodanski, "Generation of Sequential Symbolic Network Functions for Large-Scale Networks by Circuit Reduction to a Two-Port," *IEEE Trans. on Circuits and Systems — I: Fundamental Theory and Applications*, vol. 48, no. 7, July 2001.
[54] C. Pottle, *CORNAP User Manual*, School of Electrical Engineering, Cornell University, Ithaca, NY, 1968.
[55] B. Rodanski, "Computational Efficiency of Symbolic Sequential Formulae," *Proc. SMACD*, Lisbon, Portugal, pp. 45-50, Oct. 2000.
[56] P. Sannuti, N. N. Puri, "Symbolic Network Analysis — An Algebraic Formulation," *IEEE Trans. on Circuits and Systems*, vol. CAS-27, pp. 679–687, Aug. 1980.
[57] S. Seda, M. Degrauwe, W. Fichtner, "Lazy-Expansion Symbolic Expression Approximation in SYNAP," *1992 Int. Conf. Computer-Aided Design,* Santa Clara, CA, pp. 310–317, 1992.
[58] K. Singhal, J. Vlach, "Generation of Immittance Functions in Symbolic Form for Lumped Distributed Active Networks," *IEEE Trans. on Circuits and Systems*, vol. CAS-21, pp. 57–67, Jan. 1974.
[59] K. Singhal, J. Vlach, "Symbolic Analysis of Analog and Digital Circuits," *IEEE Trans. on Circuits and Systems*, vol. CAS-24, pp. 598–609, Nov. 1977.
[60] R. Sommer, "EASY — An Experimental Analog Design System Framework," *Proc. SMACD*, Paris, Oct. 1991.
[61] J. A. Starzyk, A. Konczykowska, "Flowgraph Analysis of Large Electronic Networks," *IEEE Trans. on Circuits and Systems*, vol. CAS-33, pp. 302–315, March 1986.
[62] J. A. Starzyk, J. Zou "Direct Symbolic Analysis of Large Analog Networks," *Proc. MWSCAS*, Ames, IO, pp. 421–424, Aug. 1996.
[63] M. D. Topa, "On Symbolic Analysis of Weakly-Nonlinear Circuits," *Proc. SMACD*, Kaiserslautern, Germany, Oct. 1998, pp. 207–210.
[64] J. Vlach, K. Singhal, *Computer Methods for Circuit Analysis and Design*, 2nd ed. New York: Van Nostrand Reinhold, 1994.
[65] C. Wen, H. Floberg, Q. Shui-sheng, "A Unified Symbolic Method for Steady-State Analysis of Nonlinear Circuits and Systems," *Proc. SMACD*, Kaiserslautern, Germany, Oct. 1998, pp. 218–222.

[66] G. Wierzba et al., "SSPICE — A Symbolic SPICE Program for Linear Active Circuits," *Proc. MWSCAS*, 1989.

[67] P. Wambacq, G. Gielen, W. Sansen, "A Cancellation-Free Algorithm for the Symbolic Simulation of Large Analog Circuits," *Proc. ISCAS*, San Diego, CA, pp. 1157–1160, May 1992.

[68] P. Wambacq, G. E. Gielen, W. Sansen, "Symbolic Network Analysis Methods for Practical Analog Integrated Circuits: A Survey," *IEEE Trans. on Circuits and Systems — II: Analog and Digital Signal Processing*, vol. 45, no. 10, pp. 1331–1341, Oct. 1998.

[69] Q. Yu, C. Sechen, "A Unified Approach to the Approximate Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Trans. on Circuits and Systems — I: Fundamental Theory and Applications*, vol. 43, no. 8, pp. 656–669, Aug. 1996.