Winter 1980

A COMPUTER AIDED APPROACH TO THE NOISE ANALYSIS OF RC AND OPERATIONAL AMPLIFIER NETWORKS

Donald Clarke
University of New Hampshire, Durham

THESIS

UNIVERSITY OF
NEW HAMPSHIRE

A COMPUTER AIDED APPROACH TO THE NOISE ANALYSIS OF RC AND OPERATIONAL AMPLIFIER NETWORKS

BY

DONALD CLARKE

MASTER OF SCIENCE

A COMPUTER AIDED APPROACH TO THE NOISE ANALYSIS OF RC AND OPERATIONAL AMPLIFIER NETWORKS

BY

DONALD CLARKE
B.S., University of New Hampshire, 1974

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Electrical Engineering

December, 1980

This thesis has been examined and approved.

Donald W Melvin

Thesis director, Donald W. Melvin
Associate Dean, College of Engineering and Physical Sciences
Associate Professor of Electrical and Computer Engineering

John R LaCourse

John R. LaCourse
Assistant Professor of Electrical and Computer Engineering

Walter Miller

Walter T. Miller, III
Assistant Professor of Electrical and Computer Engineering

Nov. 26, 1980

Date

ACKNOWLEDGMENTS

The author is indebted to Dr. D. W. Melvin for guidance throughout the course of this work.

The author also wishes to thank his wife Cynthia for her solid support of the effort.

Financial assistance from Northeast Electronics Corporation, Concord, N.H., is gratefully acknowledged.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A COMPUTER AIDED APPROACH TO THE NOISE ANALYSIS OF RC AND OPERATIONAL AMPLIFIER NETWORKS

by

DONALD CLARKE

University of New Hampshire, December , 1980

A noise analysis algorithm and computer program is presented which is based on a relationship between the sensitivity of a network function to variations in the value of an element within the network and the noise associated with that element. The program can handle second order RC-operational amplifier networks and will compute the output noise over the user specified bandwidth. The voltage transfer function of the network is also computed. An infinite gain multiple feedback LP filter is analyzed and the results compared with another method.

INTRODUCTION

The objective of the work presented in this thesis is to provide a means of network noise analysis that can be used without extensive prior knowledge of the subject. The need for a computer aided noise analysis capability became evident while the author was developing low noise active filters for use in telephone line noise test sets. The noise calculations were involved and time consuming.

A literature search revealed many papers dealing with analysis of network noise in operational amplifier networks. However, there were no papers on generalized noise analysis programs.

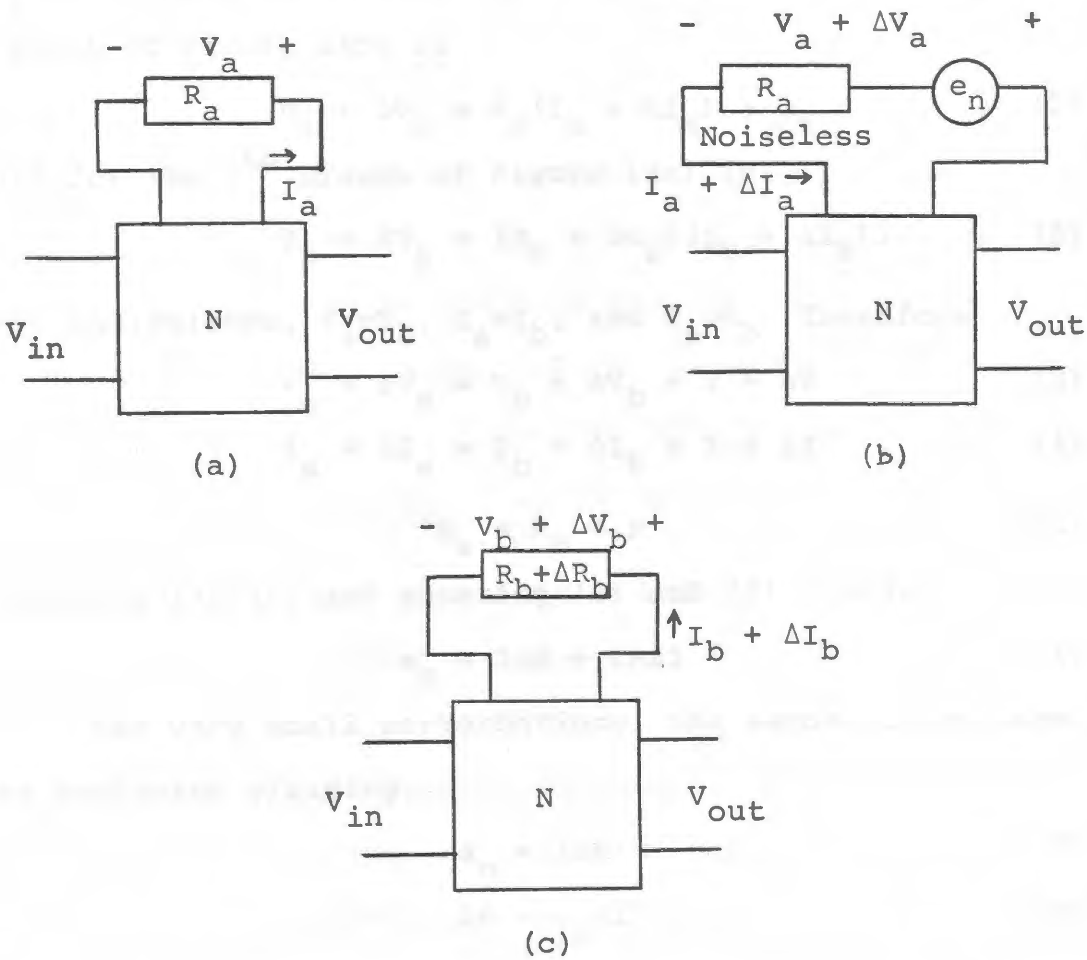
This paper presents a general noise analysis program written in Fortran that can analyze an arbitrary user specified second order RC-operational amplifier network. The input format is similar to that of well known AC analysis programs such as AC CODED. The program output consists of the network voltage transfer function and the output noise over the user specified bandwidth.

CHAPTER I

BASIC CONCEPTS AND RELATIONSHIPS

Noise-Sensitivity Relationship

The algorithm is based on a noise-sensitivity relationship described in [1] and summarized here. Consider the networks of Figure 1.



Simulation of Resistance Increment
Using a Small Voltage Source

Figure 1

Figure 1(a) is a network with a noisy resistor in the i^{th} branch. Figure 1(b) is the well known noise model of the resistor: a noiseless resistor in series with a voltage noise generator. Figure 1(c) shows the i^{th} branch with the noise of the resistor characterized as fluctuations of the value of the resistor.

The relationship between e_n, the noise generator, and R_a is derived as follows. The V-I relation for the i^{th} branch of Figure 1(b) is

$$V_a + \Delta V_a = R_a(I_a + \Delta I_a) + e_n, \quad (1)$$

and for the i^{th} branch of Figure 1(c) is

$$V_b + \Delta V_b = (R_b + \Delta R_b)(I_b + \Delta I_b). \quad (2)$$

For equivalence, $V_a=V_b$, $I_a=I_b$, and $R_a=R_b$. Therefore

$$V_a + \Delta V_a = V_b + \Delta V_b = V + \Delta V \quad (3)$$

$$I_a + \Delta I_a = I_b + \Delta I_b = I + \Delta I \quad (4)$$

$$R_a = R_b = R \quad (5)$$

Applying (3)-(5) and equating (1) and (2) yields

$$e_n = I\Delta R + \Delta R\Delta I \quad (6)$$

For very small perturbations, the second order term may be neglected yielding

$$e_n = I\Delta R \quad \text{or} \quad (7)$$

$$\Delta R = e_n/I \quad (8)$$

It has been shown that a voltage generator in series with a noiseless resistive element can be represented by a change in the resistance of that element for small variations. If the voltage source is assigned the value of the noise associated with the resistor then a noise-sensitivity relationship exists from the definition of classical sensitivity. This relationship was derived for resistive network elements because the algorithm models all network noise generators as resistors. The derivation is valid, however, for a generalized impedance element [1].

Noise Models

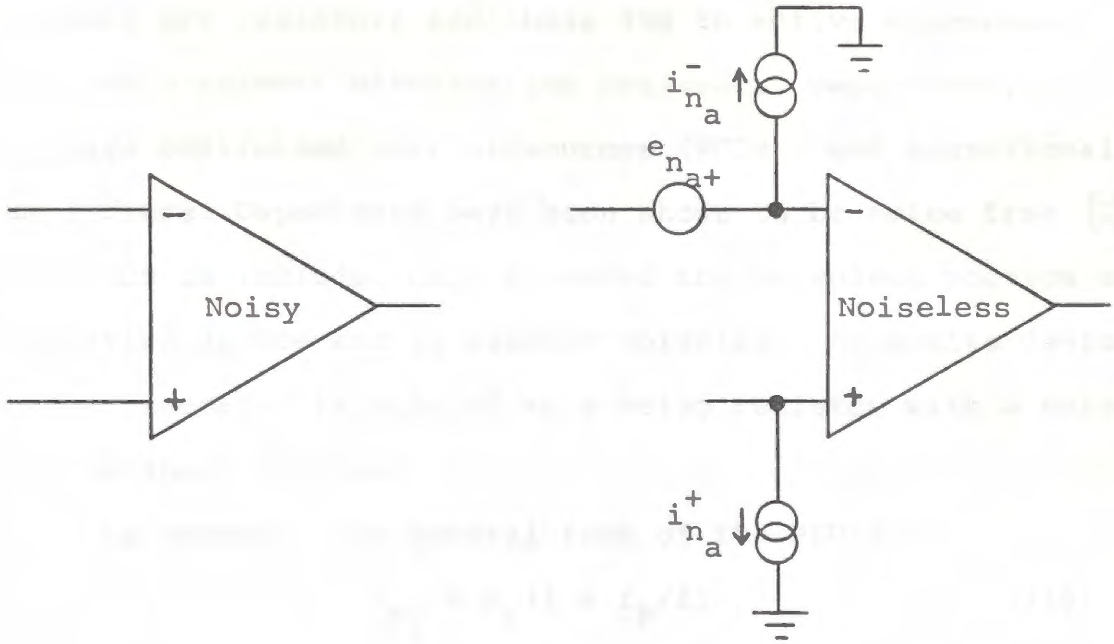
There are two sources of noise in the type of networks the algorithm can handle; thermal noise of the resistors and the noise inherent in the operational amplifiers. The program uses the power spectral density (PSD) of the noise source in its computation. The PSD is the square of the noise voltage. For resistors, the PSD is simply

$$S_{n_i} = e_n^2 = 4kTR. \qquad (9)$$

The noise of an operational amplifier is characterized by three equivalent noise generators at the input of a noiseless amplifier as shown in Figure 2.

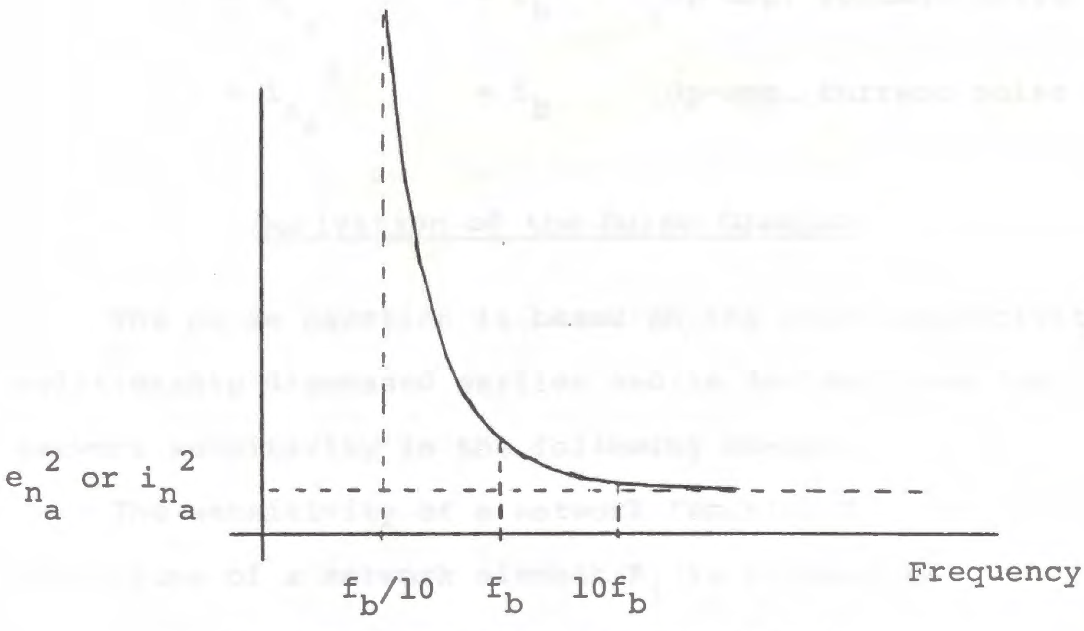
The PSD of the operational amplifier noise generators typically exhibit the 1/f characteristic shown in Figure 3 and can be fully described by the midband noise value ($e_{n_a}^2$ or $i_{n_a}^2$) usually given in V^2/Hz or A^2/Hz and the break frequency f_b [6].

As stated previously, the only noise sources in the



Operational Amplifier Noise Model

Figure 2



Op-amp Input Noise Characteristic

Figure 3

network are resistors and those due to active elements. Allowable network elements are resistors, capacitors, voltage controlled current sources (VCCS), and operational amplifiers. Capacitors have been shown to be noise free [2]. Any VCCS is included only to model the noiseless portion of an active device and is assumed noiseless. An active device noise generator is modeled as a noisy resistor with a noise PSD as shown in Figure 3.

In summary, the general form of the PSD is

$$S_{n_i} = K_o(1 + f_b/f) \qquad (10)$$

where

$K_o = 4kTR$	$f_b = 0$	Resistor noise
$= e_{n_a}^2$	$= f_b$	Op-amp. voltage noise
$= i_{n_a}^2$	$= f_b$	Op-amp. current noise

Derivation of the Noise Equation

The noise equation is based on the noise-sensitivity relationship discussed earlier and is derived from the network sensitivity in the following manner.

The sensitivity of a network function T to variations of a network element R_i is defined as

$$S_{R_i}^{T} = \frac{R_i}{T} \circ \frac{\partial T}{\partial R_i} \approx \frac{R_i}{T} \circ \frac{\Delta T}{\Delta R_i} \qquad (11)$$

for small changes. For E_{in} constant

$$\frac{\Delta T}{T} = \frac{\Delta E_{out}}{E_{out}} \quad \text{and} \quad S^{T}_{R_i} = S^{E_{out}}_{R_i} \tag{12}$$

Therefore

$$S^{T}_{R_i} = \frac{R_i}{\Delta R_i} \cdot \frac{\Delta E_{out}}{E_{out}} \tag{13}$$

From (8) and the relation $R_i = V_i/I_i$ for the i^{th} branch,

$$S^{T}_{R_i} = \frac{V_i}{e_{n_i}} \cdot \frac{\Delta E_{out}}{E_{out}} \tag{14}$$

Solving for E_{out}, the desired parameter, yields

$$\Delta E_{out} = S^{T}_{R_i} \cdot \frac{E_{out}}{V_i} \cdot e_{n_i} \tag{15}$$

Since the noise sources for all passive elements are uncorrelated, and most active element noise sources have been shown to be uncorrelated [2], the output noise power due to more than one source can be summed. This requires that (15) be written in terms of PSD's.

Since E_{out_i}, V_i, and $S^{T}_{R_i}$ are polynomials in the complex variable s, we must take the square of the magnitude of (15) to obtain the output power due to the noise generated by the i^{th} impedance element

$$S_{no_i} = (\Delta E_{out_i})^2 = \left| \left(\frac{E_{out_i}}{V_i}\right) \cdot \left(S^{T}_{R_i}\right) \right|^2 \cdot e_{n_i}^2 \tag{16}$$

e_{ni}^2 represents the PSD of the noise associated with the i^{th} impedance element and S_{no_i} represents the PSD of the output noise due to the i^{th} impedance element.

The total output noise power is $P_t = P_1 + P_2 + \ldots$ $\ldots + P_i + \ldots + P_n$ for n noise sources, where P_i is the output noise power due to the i^{th} source. The output noise PSD due to the i^{th} source is related to the noise power P_i

by

$$P_i = \int_{f_1}^{f_2} Sn_{o_i}\, df \quad (17)$$

Therefore

$$P_i = \int_{w_1}^{w_2} \left| \frac{E_{out}}{V_i} \circ S_{R_i}^{T} \right|^2 \circ K_{o_i} \circ (1 + w_{b_i}/w)\, dw \quad (18)$$

where K_o is defined in (10) and $\omega_b = 2\pi f_b$.

Summing the output power due to all sources gives

$$P_t = \sum_{i=1}^{N} \left[\int_{w_1}^{w_2} \left| \frac{E_{out}}{V_i} \circ S_{R_i}^{T} \right|^2 \circ K_{o_i} \circ (1 + w_{b_i}/w)\, dw \right] \quad (19)$$

Since $E_{n_{out}} = \sqrt{P_t}$ the noise equation becomes

$$En_o = \sqrt{\sum_{i=1}^{N} \left[\int_{w_1}^{w_2} \left| \frac{E_{out}}{V_i} \circ S_{R_i}^{T} \right|^2 \circ K_{o_i} \circ (1 + w_{b_i}/w)\, dw \right]} \quad (20)$$

The technique used to find $E_{out}(s)/V_i(s)$ is straightforward. After the network transfer function $T(s)=E_{out}(s)/E_{in}(s)$ has been found, it is necessary to find the transfer function from the input to the voltage across the i^{th} branch. Dividing $E_{out}(s)/E_{in}(s)$ by $V_i(s)/E_{in}(s)$ gives the desired result

$$\frac{E_{out}(s)/E_{in}(s)}{V_i(s)/E_{in}(s)} = \frac{E_{out}(s)}{V_i(s)} = \frac{N(s)}{N'(s)} = \frac{N(s)/D(s)}{N'(s)/D(s)} \quad (21)$$

Because the numerator and denominator are polynomials in s with coefficients that are symbol combinations and not ratios of symbol combinations, there is only one form each of them can take. Since the denominator polynomial is the same regardless of where the output is taken, the division indicated in (21) will give a ratio of two polynomials in s that is uniquely the function $E_{out}(s)/ V_i(s)$. Therefore, it is only necessary to solve for the numerator of $V_i(s)/ E_{in}(s)$.

There is still one unknown in the noise equation and that is $S_{R_i}^{T(s)}$. If the position of R_i in the transfer function is known, then $S_{R_i}^{T(s)}$ can be written down by inspection as

$$S_{R_i}^{T} = \sum_{R_i} \frac{a_{ij} s^i}{N(s)} - \sum_{R_i} \frac{b_{km} s^k}{D(s)} = \frac{AIJ(s)}{N(s)} - \frac{BKM(s)}{D(s)} \quad (22)$$

where the transfer function is coded as [3]

$$T(s) = \frac{(a_{n1}+\ldots+a_{np})s^n + (a_{(n-1)1}+\ldots+a_{(n-1)q})s^{n-1} +}{(b_{m1}+\ldots+b_{mr})s^m + (b_{(m-1)1}+\ldots+b_{(m-1)t})s^{m-1} +} \quad (23)$$

This implies that if all the R_i are coded as to their position in the transfer function, then the sensitivity function for that element can be determined easily by look up methods.

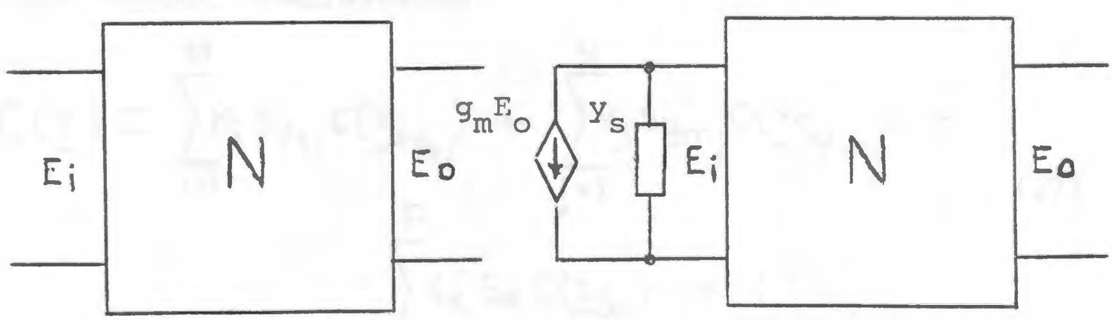
The resultant noise equation written in terms of the polynomial designators of (21) and (22) is

$$E_{n_o} = \sqrt{\sum_{i=1}^{N} \left[\int_{w_1}^{w_2} \left| \frac{N(s)}{N'(s)} \left(\frac{AIJ(s)}{N(s)} - \frac{BKM(s)}{D(s)} \right) \right|^2 K_{0_i} \left(1 + \frac{w_{b_i}}{w} \right) dw \right]} \quad (24)$$

where s = jw

The Parameter Extraction Process

The solution of the network transfer function begins with the Indefinite Admittance Matrix (IAM). The IAM represents a network whose reference or datum is external to the network. In a real circuit, ground is usually the datum node. However, the network function from one port to another is independent of which node within the network actually is the datum. To form a definite admittance matrix, a node internal to the network is taken as the reference node and its row and column are deleted from the IAM. The determinant of this matrix contains all information necessary to determine the network response provided the IAM represents a network modified as shown in Figure 4 [4].



Network Modification
Figure 4

Because the network function does not depend upon a specific reference node, any node could be chosen as the network datum. Any cofactor of the IAM, therefore, will give the same result. The notation C(Y) can be used to represent any cofactor of the IAM Y.

It has been shown [4] that if Y is the IAM of the modified network of Figure 4, and if the terms of the cofactor of the IAM [C(Y)] are sorted with respect to the symbolic parameters g_m and y_s according to

$$C(\underset{\sim}{Y}) = y_s P_{y_s} + g_m P_{g_m} + P_o \tag{25}$$

then

$$\frac{E_{out}}{E_{in}} = P_{g_m} / P_{y_s} \tag{26}$$

If some elements of the original network are left as symbols, then the coefficients of the voltage transfer function may contain these symbols or combinations of these symbols. The cofactor can be written in a form relating to these symbol combinations.

$$C(\underset{\sim}{Y}) = \sum_{i=1}^{M} K_i S_{y_{s_i}} C(\underset{\sim}{Y}_{S_{y_{s_i}}}) + \sum_{j=1}^{N} K_j S_{g_{m_j}} C(\underset{\sim}{Y}_{S_{g_{m_j}}}) + \sum_{k=1}^{P} K_k S_k C(\underset{\sim}{Y}_{S_k}) + C(\underset{\sim}{Y}_o) \tag{27}$$

where K_i, K_j, and K_k hold the sign information which resulted from the extraction of symbols,

$S_{y_{s_i}}$	is a combination of symbolic parameters containing y_s,
$S_{g_{m_j}}$	is a combination of symbolic parameters containing g_m,
S_k	is a combination of symbolic parameters containing neither g_m nor y_s,
$Y_{S_{y_{s_i}}}$	is an IAM from which the symbol set $S_{y_{s_i}}$ has been extracted and all other symbols set to zero,
$Y_{S_{g_{m_j}}}$	is an IAM from which the symbol set $S_{g_{m_j}}$ has been extracted and all other symbols set to zero, and
Y_{S_k}	is an IAM from which the symbol set S_k has been extracted and all other symbols set to zero.

We can solve for P_{g_m} and P_{y_s} of (27) by equating like terms of (25) and (27) to get

$$P_{y_s} = \frac{1}{y_s} \sum_{i=1}^{M} K_i S_{y_{s_i}} C(\underset{\sim}{Y}_{S_{y_{s_i}}}) \tag{28}$$

and

$$P_{g_m} = \frac{1}{g_m} \sum_{j=1}^{N} K_j S_{g_{m_j}} C(\underset{\sim}{Y}_{S_{g_{m_j}}}) \tag{29}$$

The following procedure is used to obtain the contribution to P_{g_m} of the j^{th} symbol combination.

<u>Step 1</u> Form the valid symbol combination $S_{g_{m_j}}$

Step 2 Extract each symbol contained in $S_{g_{m_j}}$ from the IAM using the procedure described on page 14 to form $Y_{S_{g_{m_j}}}$ keeping track of the sign information K_j

Step 3 Evaluate $C(Y_{S_{g_{m_j}}})$ to obtain a polynomial in s

Step 4 Multiply K_j by the value of each symbol contained in $S_{g_{m_j}}$ except g_m

Step 5 Combine the results of Step 3 and Step 4

Step 6 Return to Step 1 and repeat the process until all valid symbol combinations which contain g_m have been evaluated

Step 7 Sum the contributions for each power of s to obtain the numerator polynomial P_{g_m}

P_{y_s} is formed using the same procedure.

In (28) and (29) division by y_s and g_m, respectively, is indicated. This is achieved during the above process by simply ignoring them during computation.

The extraction of a symbol from the IAM is based on another form for C(Y) discussed by Alderson and Lin [4] and repeated here.

$$C(\underset{\sim}{Y}) = C(\underset{\sim}{Y}|_{\alpha=0}) + (-1)^{j+m}\alpha C(\underset{\sim}{Y}_{\alpha}) \qquad (30)$$

where α is a symbol which appears in the IAM represented by Y in exactly four elements as follows:

$$y_{ik} = \alpha + y_{ik}|\alpha=0$$

$$y_{im} = -\alpha + y_{im}|\alpha=0$$

$$y_{jk} = -\alpha + y_{jk}|\alpha=0$$

$$y_{jm} = \alpha + y_{jm}|\alpha=0$$

$Y|\alpha=0$ is an IAM where all symbols have been set to zero, and

Y_α is an IAM from which α has been extracted

The actual extraction is accomplished by adding row j to row i, adding column m to column k, then deleting row j and column m.

The relationship in (30) is extended to multiple extractions by its repeated application as illustrated by the following example. Let g_m and y_s be two symbols contained in the IAM of the modified network of Figure 4. By applying (30) once to extract g_m we get

$$C(\underset{\sim}{Y}) = C(\underset{\sim}{Y}|_{g_m=0}) + (-1)^{j_1+m_1} g_m C(\underset{\sim}{Y}_{g_m}) \tag{31}$$

Now apply (30) to this result to get

$$C(\underset{\sim}{Y}) = C(\underset{\sim}{Y}|_{g_m=0,\, y_s=0}) + (-1)^{j_1+m_1} g_m C\left[(\underset{\sim}{Y}|y_s=0)_{g_m}\right] +$$

$$(-1)^{j_2+m_2} y_s C\left[(\underset{\sim}{Y}|g_m=0)_{y_s}\right] + (-1)^{j_1+m_1}(-1)^{j_2+m_2} g_m y_s C(\underset{\sim}{Y}_{g_m y_s}) \tag{32}$$

Referring back to (25) we can equate terms to get

$$P_0 = C\left(\underset{\sim}{Y}\middle| \begin{matrix} g_m=0 \\ y_s=0 \end{matrix}\right) \tag{33}$$

$$P_{g_m} = (-1)^{j_1+m_1} g_m C\left[\left(\underset{\sim}{Y}\middle| y_s=0\right) g_m\right] \tag{34}$$

$$P_{y_s} = (-1)^{j_2+m_2} y_s C\left[\left(\underset{\sim}{Y}\middle| g_m=0\right) y_s\right] \tag{35}$$

$Y_{g_m y_s}$ does not exist. Since g_m and y_s appear in the same rows of the IAM, the extraction process results in $y_s - y_s$ in elements $y_{i_2 k_2}$ and $y_{i_2 m_2}$ meaning y_s does not appear in Y_{g_m}. The combination $g_m y_s$ is therefore invalid, and in fact can never be valid for any network.

From (26),(34),and(35)

$$\frac{E_{out}}{E_{in}} = \frac{(-1)^{j_1+m_1} C\left[(\underset{\sim}{Y}|y_s=0)_{g_m}\right]}{(-1)^{j_2+m_2} C\left[(\underset{\sim}{Y}|g_m=0)_{y_s}\right]} \tag{36}$$

This example illustrates the special case when there are no symbols in the original network.

CHAPTER II

PROGRAM DESCRIPTION

The description of the program is based on the flow diagrams of Appendix A and B. Each block of the flow diagram is numbered and referenced in the text.

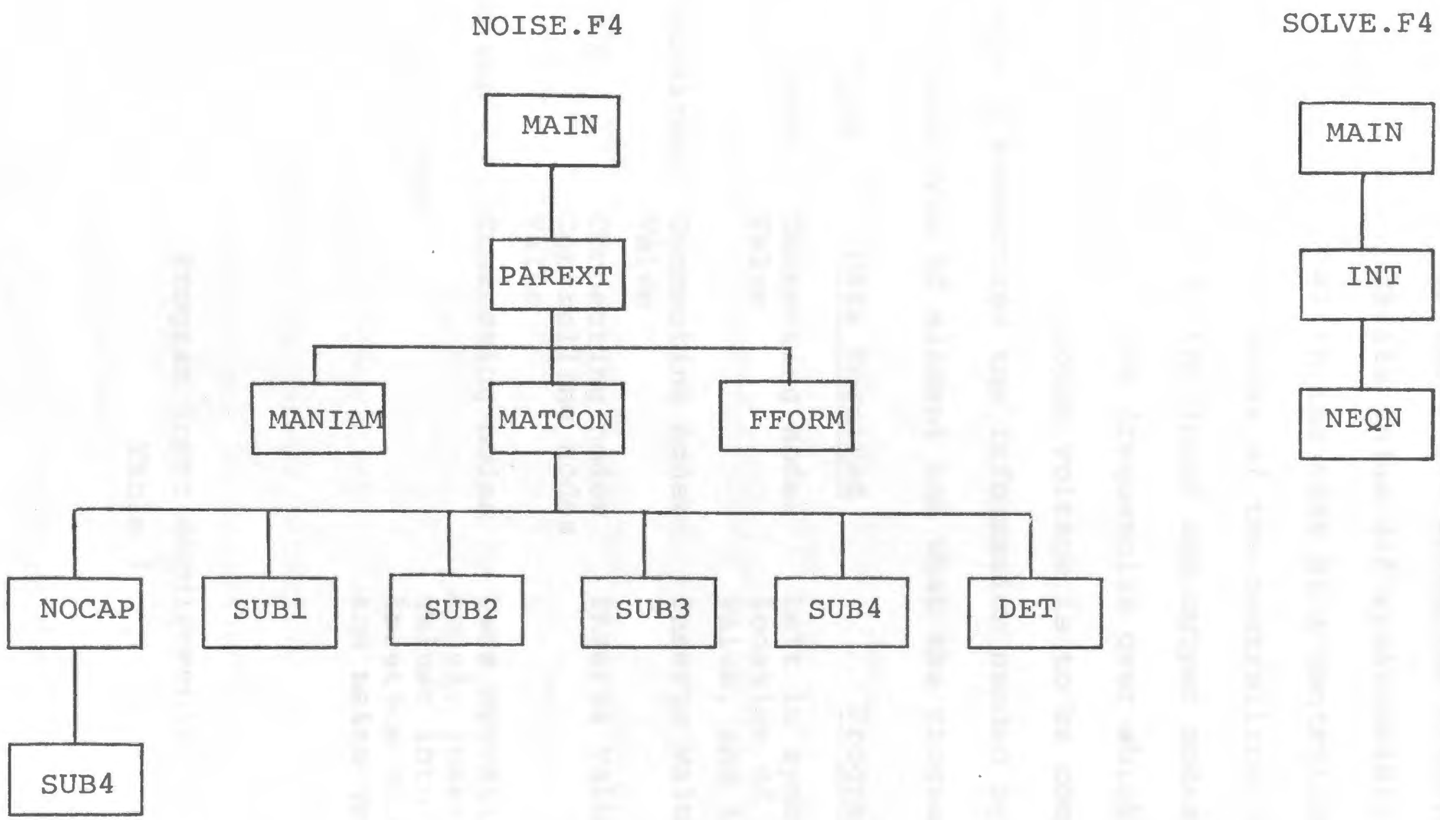
The structure of the program is diagrammed in Figure 5. There are two separate executable programs which make up the overall noise analysis package. These are NOISE.F4 and SOLVE .F4. NOISE.F4 determines the noise equation coefficients for each noise source in the network. It also determines the voltage transfer function of the network. SOLVE.F4 performs the numerical integration of each equation to determine the output noise contributed by each source then sums the result to obtain the total output noise.

NOISE.F4 consists of a main program and twelve subroutines. Figure 5 shows that the program has four levels of subroutines.

SOLVE.F4 consists of a main program and two subroutines.

Description of NOISE.F4

Blocks 1 thru 32 constitute the data entry portion of the program. Here the input data is accepted from a user specified data file and formatted so the program can process the information. The user must provide the following



Program Level Breakdown

Figure 5

information: a) the number of nodes in the network,

b) a description of each element(i.e. R,G),

c) the nodes which the element connects,

d) its value (if applicable),

e) in the case of a controlled source, the nodes of the controlling voltage,

f) the input and output nodes, and

g) the frequencies over which the output noise voltage is to be computed.

Table 1 summarizes the information needed by the program for each type of element and what the program does with it.

Type	Data Required	Program Action
Resistor	Connecting nodes Value	Left in symbol form. Stores location of symbol, its value, and its PSD
Capacitor	Connecting nodes Value	Inserts value into IAM
VCCS	Connecting nodes Controlling nodes Value	Inserts value into IAM
Op-amp	Connecting nodes	Uses operational amplifier model. Inserts non-symbol values into IAM. Stores location of symbols, values, and noise generator PSD's.

Program Input Requirements
Table 1

Block 33 provides a check on the order of the network IAM. If the order of the IAM is less than three, the parameter extraction method cannot be used. If the order is equal to three, then the algorithm can be used but no symbolic parameters can be used except g_m and y_s which are necessary to solve for the voltage transfer function. If the order of the IAM is greater than three, the algorithm can be used.

Whenever a symbol is extracted from the IAM, a row and column are deleted, reducing the order of the IAM by one. We desire to solve C(Y), but to have a cofactor the order of Y must be greater than one. Since we are looking for the voltage transfer function, each term must contain either g_m or y_s(See (25) and (26)). Both symbols will not appear in the same term. If N is the order of the IAM, then N-2 is the maximum number of extractions that can be performed. However, one of these is g_m or y_s. Therefore only N-3 original network symbolic parameters can be processed at any one time.

It is easy to see that if N=3 then no original network symbolic parameters can be extracted. But if N>3 then the symbolic parameters can be processed N-3 at a time. Obviously, if N<3 not even g_m or y_s can be symbols thus defeating the algorithm.

In block 34 the frequencies over which the output noise is to be computed are requested. At this point, the network has been fully defined and the processing may begin.

Block 35 segments the symbols into groups of N-3. Because of array dimensioning constraints, the maximum number of symbols that can be processed at any one time is three. If N-3 is greater than three, the group size is set at three. Symbols not belonging to the group being processed have their values inserted into the IAM(Block 36).

In block 37, the network is modified as shown in Figure 4. This procedure involves storing the position of the elements in the IAM where g_m and y_s appear. This is the same procedure indicated in the right-hand column of Table 1 for resistor symbols. The coding of the positions of the symbolic parameters in the IAM will be discussed next.

Each of the network elements will appear in the IAM in four places as illustrated by the example for R_1 and C_1 in Figure 6.

$$\begin{bmatrix} C_1 & 0 & -C_1 \\ 0 & 0 & 0 \\ -C_1 & 0 & C_1 \end{bmatrix} s + \begin{bmatrix} G_1 & -G_1 & 0 \\ -G_1 & G_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Symbol Location in IAM

Figure 6

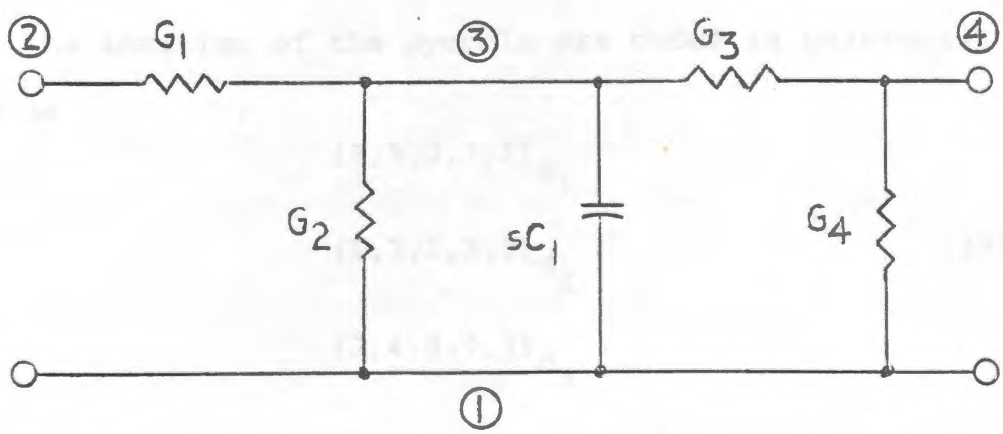
The symmetrical location of each element in the IAM is the result of applying Kirchhoff's current law at each node of the network to obtain the network nodal equations in the form I = YV, where Y = sC + G, the IAM.

The rows and columns where each element is located in the IAM can be represented by a quintuple (5 element vector)

$$(i\ j\ k\ m\ p) \tag{37}$$

where i and j are row indicators, k and m are column indicators, and p is the symbol identifier. Each symbol is coded in this manner [4].

Blocks 38 and 39 involve forming a symbol combination and checking its validity (e.g., whether it is part of the voltage transfer function). This can be done without touching the IAM itself by comparing the elements of the quintuples. The process is described in detail in [4], and is summarized here. It is best illustrated by example. Consider the network



Low Pass Filter Network

Figure 7

where G_1, G_2, and G_3 are noise sources and therefore symbols. The IAM associated with this network is

$$\begin{bmatrix} sC_1+G_2+G_4 & 0 & -sC_1-G_2 & -G_4 \\ 0 & G_1 & -G_1 & 0 \\ -sC_1-G_2 & -G_1 & G_1+G_2+G_3+sC_1 & -G_3 \\ -G_4 & 0 & -G_3 & G_3+G_4 \end{bmatrix} \tag{38}$$

The location of the symbols are coded in quintuplet form as

$$\begin{aligned} &(2,3,2,3,1)_{G_1} \\ &(1,3,1,3,2)_{G_2} \\ &(3,4,3,4,3)_{G_3} \end{aligned} \tag{39}$$

For this example, the network will not be modified as the procedure can be established without it.

A symbol combination such as G_1G_2 is invalid if G_2 does not appear in the resultant IAM after the symbol G_1 has been extracted. The procedure to extract a symbol involves adding row j to row i, adding column m to column k, then deleting row j and column m. Consider the symbol combination G_1G_2. By extracting G_1, the IAM becomes

$$\begin{bmatrix} sC_1+G_2+G_4 & -sC_1-G_2 & -G_4 \\ -sC_1-G_2 & G_2+G_3+sC_1 & -G_3 \\ -G_4 & -G_3 & G_3+G_4 \end{bmatrix} \tag{40}$$

The position of G_2 in this IAM is

$$(1,2,1,2,2) \tag{41}$$

Since G_2 exists in (40), the combination G_1G_2 is valid.

Now consider the symbol combination G_1G_3. After extracting G_1, the position of G_3 in the resulting IAM is

$$(2,3,2,3,3) \tag{42}$$

This combination is also valid.

The process continues for the combination $G_1G_2G_3$ by extracting G_2 from (40). The resulting IAM is

$$\begin{bmatrix} G_3+G_4 & -G_3-G_4 \\ -G_3-G_4 & G_3+G_4 \end{bmatrix} \tag{43}$$

The symbol combination $G_1G_2G_3$ is valid since G_3 exists in (43) after G_1 and G_2 have been extracted. The quintuple for G_3 in (43) is

$$(1,2,1,2,3) \tag{44}$$

Further insight into the procedure can be gained by utilizing a matrix L to organize the operation. The procedure is as follows. First, insert all quintuples into the first column of $\underset{\sim}{L}$ as shown in (45).

$$\begin{bmatrix} (2\ 3\ 2\ 3\ 1) \\ (1\ 3\ 1\ 3\ 2) \\ (3\ 4\ 3\ 4\ 3) \end{bmatrix} \tag{45}$$

When a valid combination is found, the associated quintuple is inserted into the next higher column of L. For the example given, (41) and (42) would be inserted into column two of L as shown in (46).

(44) reveals that G_3 exists in the resultant IAM after both G_1 and G_2 have been extracted. This is indicated by

$$\begin{bmatrix} (2\ 3\ 2\ 3\ 1) & (1\ 2\ 1\ 2\ 2) \\ (1\ 3\ 1\ 3\ 2) & (2\ 3\ 2\ 3\ 3) \\ (3\ 4\ 3\ 4\ 3) & \end{bmatrix} \tag{46}$$

inserting (44) into the third column of L as shown in (47)

$$\begin{bmatrix} (2\ 3\ 2\ 3\ 1) & (1\ 2\ 1\ 2\ 2) & (1\ 2\ 1\ 2\ 3) \\ (1\ 3\ 1\ 3\ 2) & (2\ 3\ 2\ 3\ 3) & \\ (3\ 4\ 3\ 4\ 3) & & \end{bmatrix} \tag{47}$$

For this example, the process would continue, checking the validity of G_1G_3, G_1, G_2, and G_3.

Block 40 involves the actual extraction of the symbols from the IAM. First, the original IAM must be saved since extraction will destroy the original matrix information. Once this has been done, the adding and deleting of rows and columns can begin. In the example, the information necessary to direct the extraction process is available in (47) for the symbol combination $G_1G_2G_3$. (47) is rewritten here with reference pointers added to clarify the discussion.

$$\begin{bmatrix} \rightarrow(2\ 3\ 2\ 3\ 1) & \rightarrow(1\ 2\ 1\ 2\ 2) & \rightarrow(1\ 2\ 1\ 2\ 3) \\ (1\ 3\ 1\ 3\ 2) & (2\ 3\ 2\ 3\ 3) & \\ (3\ 4\ 3\ 4\ 3) & & \end{bmatrix} \tag{48}$$

If G_1, indicated by the reference pointer in column one, is extracted from the IAM, the resulting IAM has symbols G_2 and G_3 whose positions are given by

$$\begin{gathered} (1,2,1,2,2) \text{ and} \\ (2,3,2,3,3) \end{gathered} \tag{49}$$

If G_2, indicated by the reference pointer in column two, is extracted from the IAM which already has had G_1 extracted, the resulting IAM has symbol G_3 in the positions indicated by

$$(1,2,1,2,3) \tag{50}$$

The three quintuples (49) and (50) contain the row and column information necessary to perform the actual extraction of G_1, G_2, and G_3 from the IAM.

Starting in column one, G_1 is extracted from the original IAM of order four by adding row 3 to row 2, adding column 3 to column 2, then deleting row 3 and column 3. The IAM is now of order three. Then, following the pointer in column two of L, G_2 is extracted from the new IAM by adding row 2 to row 1, adding column 2 to column 1, then deleting row 2 and column 2. The IAM is now second order. The last extraction is accomplished on the resultant IAM by adding row 2 to row 1, adding column 2 to column 1, then deleting row 2 and column 2. The final IAM is of order one.

When performing the elementary row and column operation operations, the sign may change. It is important to retain the sign information. The sign term will be $(-1)^{j+m}$ for each extraction.

In block 41, the dimension of the resultant IAM is reduced by one to obtain the matrix whose determinant is the desired cofactor. This matrix, which is of the general form sC + G, is then converted using equivalence transformations to the form K(sI - A) to take advantage of one of many [4] fast efficient algorithms to solve for the characteristic equation of the matrix A. The process is discussed in detail in [4] and summarized here.

Starting with an nxn matrix of the form sC+G, perform the following steps:

Step 1

Perform the elementary row and column operations to convert the matrix into the form of (51). If the rank of C is equal to n, then the form is sI-A and the process is complete.

$$P_1(sC + G)Q_1 = \begin{bmatrix} I_{i\times i} & 0 \\ 0 & 0 \end{bmatrix} s + \begin{bmatrix} G_{11_{i\times i}} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \qquad (51)$$

The final result is

$$\text{Det}\left[s\underset{\sim}{C} + \underset{\sim}{G}\right] = \frac{1}{\text{Det}\left[\underset{\sim}{P}_1\underset{\sim}{Q}_1\right]} \text{Det}\left[s\underset{\sim}{I} + \underset{\sim}{P}_1\underset{\sim}{G}\underset{\sim}{Q}_1\right] \qquad (52)$$

If i=0, then

$$\text{Det}\left[s\underset{\sim}{C} + \underset{\sim}{G}\right] = \frac{1}{\text{Det}\left[\underset{\sim}{P}_1\underset{\sim}{Q}_1\right]} \text{Det}\left[\underset{\sim}{P}_1\underset{\sim}{G}\underset{\sim}{Q}_1\right] \qquad (53)$$

is the final result. If i≠n and i≠0, then proceed to step 2.

Step 2

Perform the elementary row and column operations to convert (51) into the form of (54).

$$\underset{\sim}{P}_2\underset{\sim}{P}_1\left[s\underset{\sim}{C} + \underset{\sim}{G}\right]\underset{\sim}{Q}_1\underset{\sim}{Q}_2 =$$

$$\begin{bmatrix} I_{i\times i} & 0 \\ 0 & 0 \end{bmatrix} s + \begin{bmatrix} G_{11_{i\times i}} & 0 & G_{13} \\ 0 & I_{j\times j} & 0 \\ G_{31} & 0 & 0 \end{bmatrix} \qquad (54)$$

If the rank of G_{22} of (51) is equal to n-i, then the form is sI-A and the process is complete.

$$\mathrm{Det}\left[s\underset{\sim}{C}+\underset{\sim}{G}\right] = \frac{1}{\mathrm{Det}\left[\underset{\sim}{P}_2\underset{\sim}{P}_1\underset{\sim}{Q}_1\underset{\sim}{Q}_2\right]}\mathrm{Det}\left[s\underset{\sim}{I}+\underset{\sim}{G}_{11}\right] \qquad (55)$$

is the final result. G_{11} is as depicted in (54). If $i+j\neq n$, then proceed to step 3.

Step 3

Perform the elementary row and column operations to convert (54) into the form of (56).

$$\underset{\sim}{P}_3\underset{\sim}{P}_2\underset{\sim}{P}_1\left[s\underset{\sim}{C}+\underset{\sim}{G}\right]\underset{\sim}{Q}_1\underset{\sim}{Q}_2\underset{\sim}{Q}_3 =$$

$$\begin{bmatrix} \underset{q\times p}{C_{11}} & C_{12} & 0 & 0 & 0 \\ C_{21} & \underset{v\times t}{C_{22}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} s + \begin{bmatrix} \underset{q\times p}{G_{11}} & G_{12} & 0 & \underset{q\times p}{I} & 0 \\ G_{21} & \underset{v\times t}{G_{22}} & 0 & 0 & 0 \\ 0 & 0 & I_{j\times j} & 0 & 0 \\ I_{p\times p} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0_{r\times u} \end{bmatrix} \qquad (56)$$

$v = i-q \qquad r = k-p$

$t = i-p \qquad u = k-q$

where $k=n-(i+j)$. If $p<k$ or $q<k$, then $\det(sC+G)=0$. If $p=q=k$, then the result is given by (57).

$$\mathrm{Det}\left[s\underset{\sim}{C}+\underset{\sim}{G}\right] = \frac{(-1)^{k^2}}{\mathrm{Det}\left[\underset{\sim}{P}_3\underset{\sim}{P}_2\underset{\sim}{P}_1\underset{\sim}{Q}_1\underset{\sim}{Q}_2\underset{\sim}{Q}_3\right]}\mathrm{Det}\left[s\underset{\sim}{C}_{22}+\underset{\sim}{G}_{22}\right] \tag{57}$$

where C_{22} and G_{22} are as depicted in (56). If C_{22} in (57) is the identity matrix, then the form is sI-A and the process is complete. If $C_{22} \neq I$, then we return to step 1 starting with $sC_{22} + G_{22}$ of (57) instead of sC + G. The process is iterated until the desired form is achieved.

Block 42 represents the algorithm for solving det(sI-A). An existing program [5] was adapted for this application so that the output vector of the coefficients of the characteristic polynomial would be in the proper form. It is also possible for the matrix obtained in block 41 to take the form G. This will occur if the original network had no capacitors, or if the capacitor values were eliminated by the parameter extraction process. For this case, it is impossible to convert the form of the matrix to sI-A so a separate subroutine called NOCAP is used to solve det(G).

Blocks 43 thru 45 code the transfer function as described on page 9. When all N-3 symbols in the group being processed have been processed, the transfer function is complete.

In block 46, a symbol in the group of N-3 is selected to begin the process of determining its contribution to the output noise. Block 47 selects the nodes of the chosen symbol as the new network output port and modifies the network with g_m and y_s accordingly. This follows the same procedure as block 37.

Blocks 48 thru 55 are similar to blocks 38 thru 45 except that only the numerator polynomial of the transfer function is computed since, according to (21), the denominator polynomial is not required to form the noise equation. Care must be taken not to destroy the coding of the positions of the symbols in the original transfer function.

In block 56, the sensitivity function for the symbol being processed is formed from the coded original transfer function according to (22) and (23).

In block 57, the coefficients of the noise equation are computed from the polynomials N(s)= PGMOUT(i), D(s)= PYSOUT(i), N'(s)= PGMJ(i), AIJ(s)= AIJ(i), and BKM(s)= BKM(i) according to (24).

In block 58 thru 60 the noise equation coefficients, the PSD information for the source being processed, and the frequency limits over which the noise is to be computed are written on a user specified data file. The program then sequences through the remaining symbols in the group. When all the symbols of a group have been processed, another group is selected and the process from block 36 thru 60 is repeated; continuing until all symbols have been processed.

Finally, in block 61, the transfer function is printed along with a message instructing the user to execute SOLVE. Included in this message is the name of the data file which contains the noise equation information.

Description of SOLVE.F4

Block 1 constitutes the data entry portion of the program. Here the noise equation coefficients, the noise source PSD, and the frequency range of integration are read from the user specified data file.

Blocks 2 thru 6 perform the actual integration of the noise equation. The integration routine uses the trapezoidal rule method of numerical integration with correction terms generated using Romberg's Method. It was adapted from DQATR which is part of the IBM Scientific Subroutine Package.

Blocks 7 and 8 keep a running sum of the output noise power as each source is evaluated.

When all sources(symbols) have been evaluated, block 9 takes the square root of the noise power to obtain the output noise voltage.

The output noise voltage is printed in block 10.

CHAPTER III

ANALYSIS OF AN INFINITE GAIN MULTIPLE FEEDBACK LOW PASS FILTER

An infinite gain multiple feedback low pass filter will be analyzed to illustrate the procedure. The circuit is shown in Figure 8 . This is the same circuit discussed by Treleaven et al [6] . The results of the two methods will be compared and discussed in the next chapter.

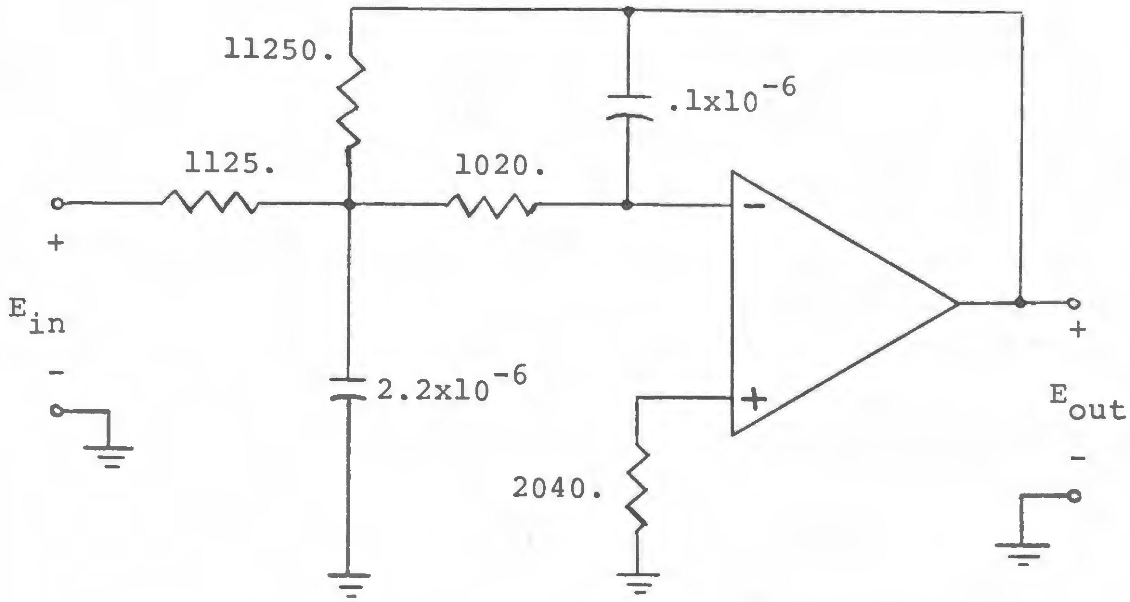
Step 1

Label the nodes of the network with ground as node 1 as shown in Figure 9 .

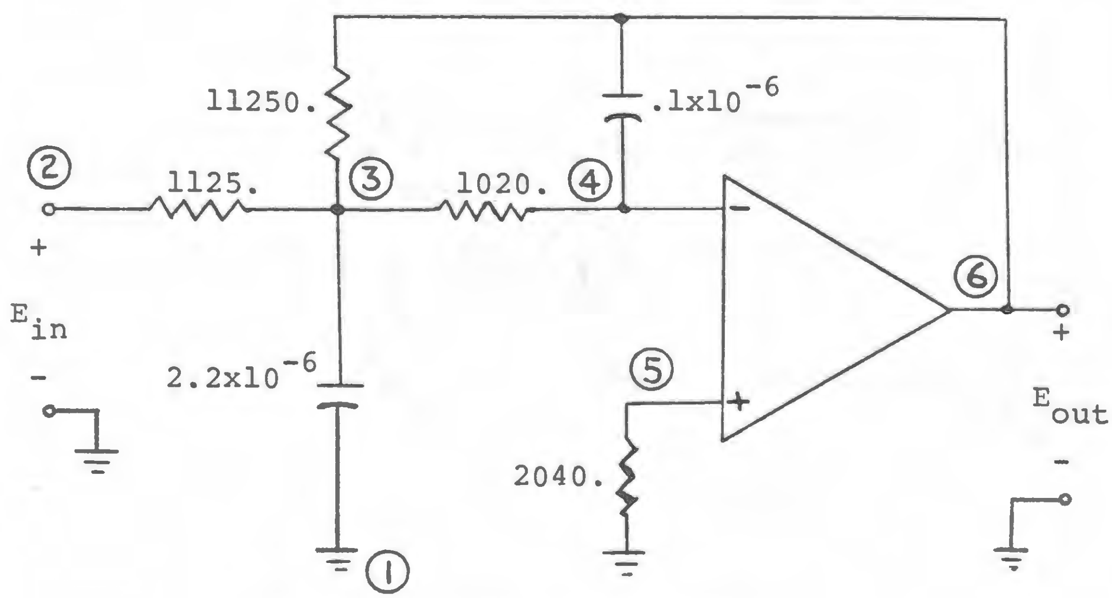
Step 2

Create a data file with the network elements entered in the format of Table 2 . The discussion assumes the reader has knowledge of Fortran and the DEC-system 10 and has already performed the login procedure. With the system in Monitor mode(indicated by a "."), type the following. (User entries are underscored)

```
.CREATE EX.DAT
*00100   6
 00200   R
 00300   2,3
 00400   1125.
 00500   C
```

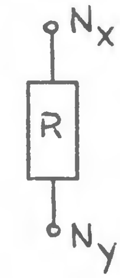


Infinite Gain Multiple Feedback LP Filter
Figure 8



LP Filter with Nodes Labeled
Figure 9

Resistor



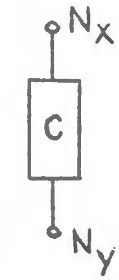
Entered in data file using three separate lines.

Type

Nodes(From,To): (N_x, N_y) or (N_y, N_x)

Value

Capacitor



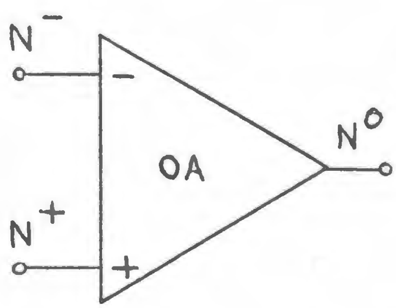
Entered in data file using three lines.

Type

Nodes(From,To): (N_x, N_y) or (N_y, N_x)

Value

Operational Amplifier

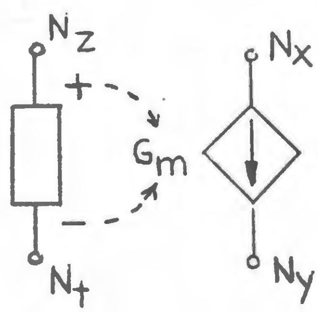


Entered in data file using two lines.

Type

Nodes(-in,+in,out): (N^-, N^+, N^o)

Voltage Controlled Current Source



Entered in data file using four lines.

Type

Controlled nodes (From,To): (N_x, N_y)

Controlling nodes (From,To): (N_z, N_t)

Value

Circuit Element Models

Table 2

```
00600   1,3
00700   2.2D-6
00800   R
00900   3,6
01000   11250.
01100   R
01200   3,4
01300   1020.
01400   C
01500   4,6
01600   .1D-6
01700   R
01800   1,5
01900   2040.
02000   OA
02100   4,5,6
02200
02300   1,2
02400   1,6
02500   1.
02600   10000.
02700   $
*B
 EXIT
.
```

When entering the network data, the user must type a TAB or CTRL I for each line entry in the file. This will position the data in column 7. Also, the $ indicates the user has typed ESC. Each line is terminated by RETURN. The asterisk indicates the system is in the editor mode. Exiting the editor via the B command deletes the line numbers from the file.

Step 3

Execute the noise analysis program by typing the following.

```
.EX NOISE

LINK: Loading

(LNKXCT NOISE Execution)

INPUT FILENAME = CKT1.DAT

OUTPUT FILENAME = SYNDI

                         VOLTAGE TRANSFER FUNCTION

                   S^4       S^3       S^2       S

NUMERATOR   = 0.00D+0 0.00D+0 0.00D+0 0.14D-3 -0.40D+7
DENOMINATOR = 0.00D+0 0.00D+0 0.10D+1 0.89D+3  0.40D+6

+++EXECUTE PROGRAM CALLED SOLVE AND USE SYNDI AS THE
FILENAME+++

STOP

END OF EXECUTION

CPU TIME: 1.89  ELAPSED TIME: 51.68

EXIT

.EX SOLVE
```

```
LINK: Loading

(LNKXCT SOLVE Execution)

INPUT FILENAME = SYNDI

THE OUTPUT NOISE =      0.856D-05   VOLTS

STOP

END OF EXECUTION

CPU TIME: 14.38 ELAPSED TIME: 22.92

EXIT

.
```

CHAPTER IV

DISCUSSION & CONCLUSIONS

The transfer function printed in Chapter III for the Infinite Gain Multiple Feedback Low Pass Filter is the correct one for the network analyzed. However, it is not the ideal transfer function

$$T(s) = \frac{k_0}{s^2 + k_1 s + k_2}$$

that might be expected. Due to the finite gain, finite input resistance, and non-zero output resistance of the op-amp model used, extra terms appear in the transfer function. The algorithm requires that the op-amp have a finite gain though it may be very high. It also requires that the model have a finite output resistance. These parameters are simulated using voltage controlled current sources and resistors since they both are of the form sC+G. If the gain or output resistor were not connected across the appropriate current source, the current would have no path to follow and the model would break down.

Comparison of the results of this work with that of Treleaven et al [6] shows a disagreement of less than 3% in the value of the output noise. The effects of the non-ideal op-amp model appear to be minimal.

One of the problems with computer-aided circuit analysis is keeping the numbers within the range that can be represented by the hardware. Frequency scaling and transfer function normalization are employed by the program to help minimize the effects of this type of machine limitation.

When capacitors are read from the input data file, their values are multiplied by $1x10^9$ before they are inserted into the IAM. After the transfer function has been formed, it is normalized so the coefficient of the highest power of s in the denominator is 1.0. Then the transfer function is unscaled by multiplying each term s^k by $(10^{-9})^k$. Subsequent to unscaling, the transfer function is again normalized so the coefficient of the highest power of s in the denominator is 1.0. This unscaled and twice normalized transfer function is the one printed by the program.

The numerator and denominator of the noise equation are each of the form

$$k_0 + k_1\omega + k_2\omega^2 + k_3\omega^3 + k_4\omega^4 + k_5\omega^5 + k_6\omega^6 + k_7\omega^7 + k_8\omega^8 \qquad (58)$$

which can be written

$$k_0 + k_1 w + (k_2^{1/2} w)^2 + (k_3^{1/3} w)^3 + (k_4^{1/4} w)^4 + (k_5^{1/5} w)^5 + (k_6^{1/6} w)^6 + (k_7^{1/7} w)^7 + (k_8^{1/8} w)^8 \qquad (59)$$

It is necessary to implement the noise equation in the form of (59) to avoid exponent overflows when ω^k is evaluated. Multiplying ω by the k^{th} root of the coefficient then raising the product to the k^{th} power relieves the problem to a sufficient extent.

The warning "EXPONENT UNDERFLOW" is printed during program execution when larger networks are analyzed. This means that the result of a multiplication or a division had a negative exponent whose magnitude was greater than that which can be represented by the machine. This occurred for the example of Chapter III. The accuracy of that analysis implies that the effect of the underflows is minimal.

Conclusions

The output noise voltage of 8.56μV obtained in Chapter III is within 3% of the 8.32μV value obtained by Treleaven et al [6]. The error can be accounted for by the slight difference in the model used for the operational amplifier current noise generators. The model used in this work is

$$K(1 + f_b/f) \tag{60}$$

while [6] uses

$$K(f_b/f) \tag{61}$$

There will be a larger contribution to the output noise due to the current noise generators when the model of (60) is used.

The excellent agreement between the results of the two methods supports the validity of the algorithm implemented in this work.

Topics for Future Investigation

The program presented in this work is limited to second order networks containing resistors, capacitors, and operational amplifiers. The algorithm has been proven valid and could be expanded to include other network elements such as inductors and transistors. Any element can be handled by the program if it is modeled in the form sC+G.

The implementation uses symbolic functions and evaluates them using a numerical integration technique. This was found to be extremely slow taking several minutes for moderate size networks. Alternative approaches which avoid the use of numerical integration might be more desirable from a cost/time standpoint and should be investigated. To be useful, however, it must be able to handle an arbitrary user defined network.

REFERENCES

(1) A. G. J. Holt and M. R. Lee," A Relationship Between Sensitivity and Noise," International Journal of Electronics, pp. 591-594, 1969

(2) D. H. Treleaven," Electrical Noise in Inductorless Filters," PhD Thesis, University of Calgary, Calgary, Alberta, Canada, pp. 125-132, 1972

(3) C. F. Yokomoto," A Simple Bookkeeping Scheme for Computing Sensitivities of Symbolic Transfer Functions," IEEE Trans. Circuits and Systems, Vol. CAS-21, pp. 606-608, Sept. 1974

(4) G. E. Alderson and P. M. Lin," Computer Generation of Symbolic Network Functions - A New Theory and Implementation," IEEE Trans. Circuit Theory, Vol. CT-20, pp. 48-56, Jan. 1973

(5) D. E. McLaughlin," A Computer Oriented Course in Linear Algebra," Augustana College, 1971

(6) F. N. Trofimenkoff, D. H. Treleaven, and L. T. Bruton, " Noise Performance of RC-Active Quadratic Filter Sections," IEEE Trans. Circuit Theory, Vol. CT-20, pp. 524-532, Sept. 1973

APPENDIXES

APPENDIX A

OPERATIONAL AMPLIFIER MODEL

The model used for all operational amplifiers is shown in Figure 10 on page 46 . The resistors Re_{n_a} , $Ri_{n_a}^-$, and $Ri_{n_a}^+$ represent the input equivalent noise generated by the op-amp. The values for $Ri_{n_a}^-$ and $Ri_{n_a}^+$, the current noise sources, were chosen to be ten times larger than the largest expected network resistance. The PSD assigned to these resistors is the PSD of the op-amp equivalent input current noise generators and is not in any way related to the resistor value. The value for Re_{n_a} was chosen to be ten times smaller than the smallest expected network resistance. The PSD assigned to this resistor is the PSD of the op-amp equivalent input voltage noise generator and is not related to the resistor value.

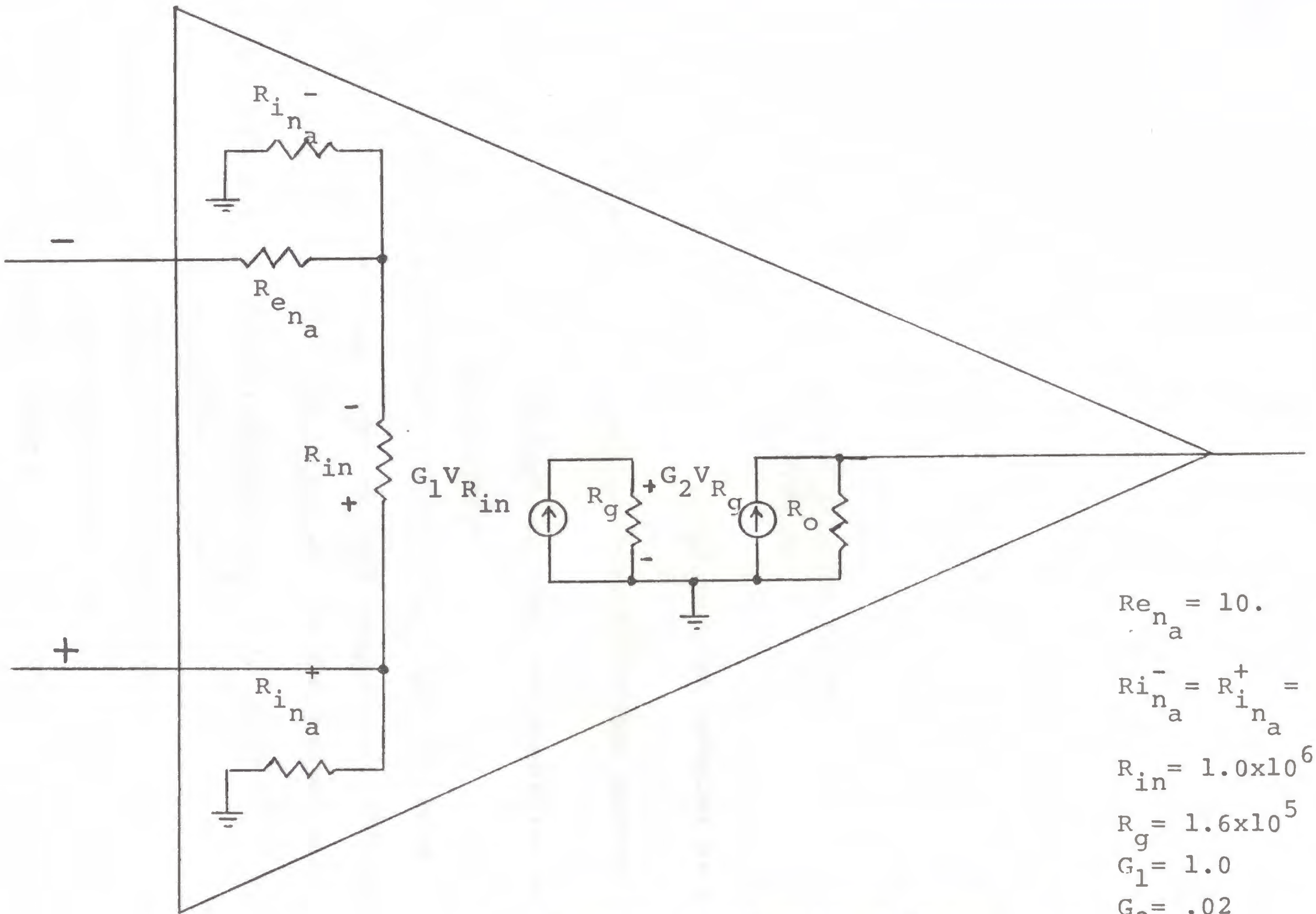
The noise generator PSD's are of the form $K(1+\omega_b/\omega)$ where

$$K = 7\text{x}10^{-17}\ V^2/\text{radian} \quad \text{voltage noise generator}$$

$$2.39\text{x}10^{-9}\ V^2/\text{radian} \quad \text{current noise generator}$$

and $\omega_b = 785.4$ radians/second

These values are derived from typical values for $e_{n_a}^2$, $i_{n_a}^2$, and f_b given in [6] as follows. The three source equivalent input noise model of an operational amplifier is shown in Figure 11. The algorithm cannot handle an ideal



$Re_{n_a} = 10.$

$Ri_{n_a}^{-} = R_{i_{n_a}}^{+} = 10^8$

$R_{in} = 1.0\text{x}10^6$

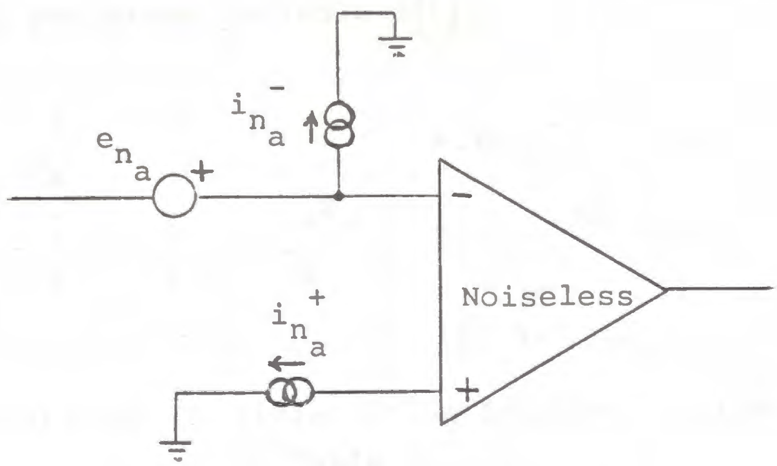
$R_g = 1.6\text{x}10^5$

$G_1 = 1.0$

$G_2 = .02$

$R_o = 50.$

Operational Amplifier Model
Figure 10



Operational Amplifier Noise Model
Figure 11

voltage source such as e_{n_a}, so it must be modeled as a noiseless resistor R in series with a voltage source e_{n_a}. The value of R should be kept as small as possible. A 10 ohm resistor is used in the program.

Ideal current sources such as $i_{n_a}^-$ and $i_{n_a}^+$ pose a similar problem. They also are not compatible with the algorithm so they are modeled as a noiseless resistor R in parallel with a current source i_{n_a}. The value of R should be kept as large as possible. A 100M ohm resistor is used in the program. The algorithm, however, cannot handle current noise sources so they must be converted to equivalent voltage noise sources. This is accomplished by multiplying i_{n_a} by R to get e_{n_i}.

The typical values of $e_{n_a}^2$, $i_{n_a}^{-2}$, $i_{n_a}^{+2}$, and f_b for the

741 op-amp are given in Table 3[6].

$e_{n_a}^2$	4.4×10^{-16} V^2/Hz
$i_{n_a}^2 = i_{n_a}^{-2} = i_{n_a}^{+2}$	1.5×10^{-24} A^2/Hz
f_b	125 Hz

Operational Amplifier Noise Characteristics

Table 3

Since the program uses ω instead of f as the frequency variable, $e_{n_a}^2$ and $i_{n_a}^2$ must be divided by 2π and f_b multiplied by 2π to give the correct result.

Applying the appropriate factors above to $e_{n_a}^2$, $i_{n_a}^2$, and f_b yields the values for K and ω_b appearing on page 45 .

The remaining elements of the operational amplifier model of Figure 10 are used to simulate a finite input resistance(R_{in}), a finite frequency invariant gain(R_g), and a finite output resistance(R_o). The values chosen for these resistors reflect a typical 741. The parameters are summarized in Table 4.

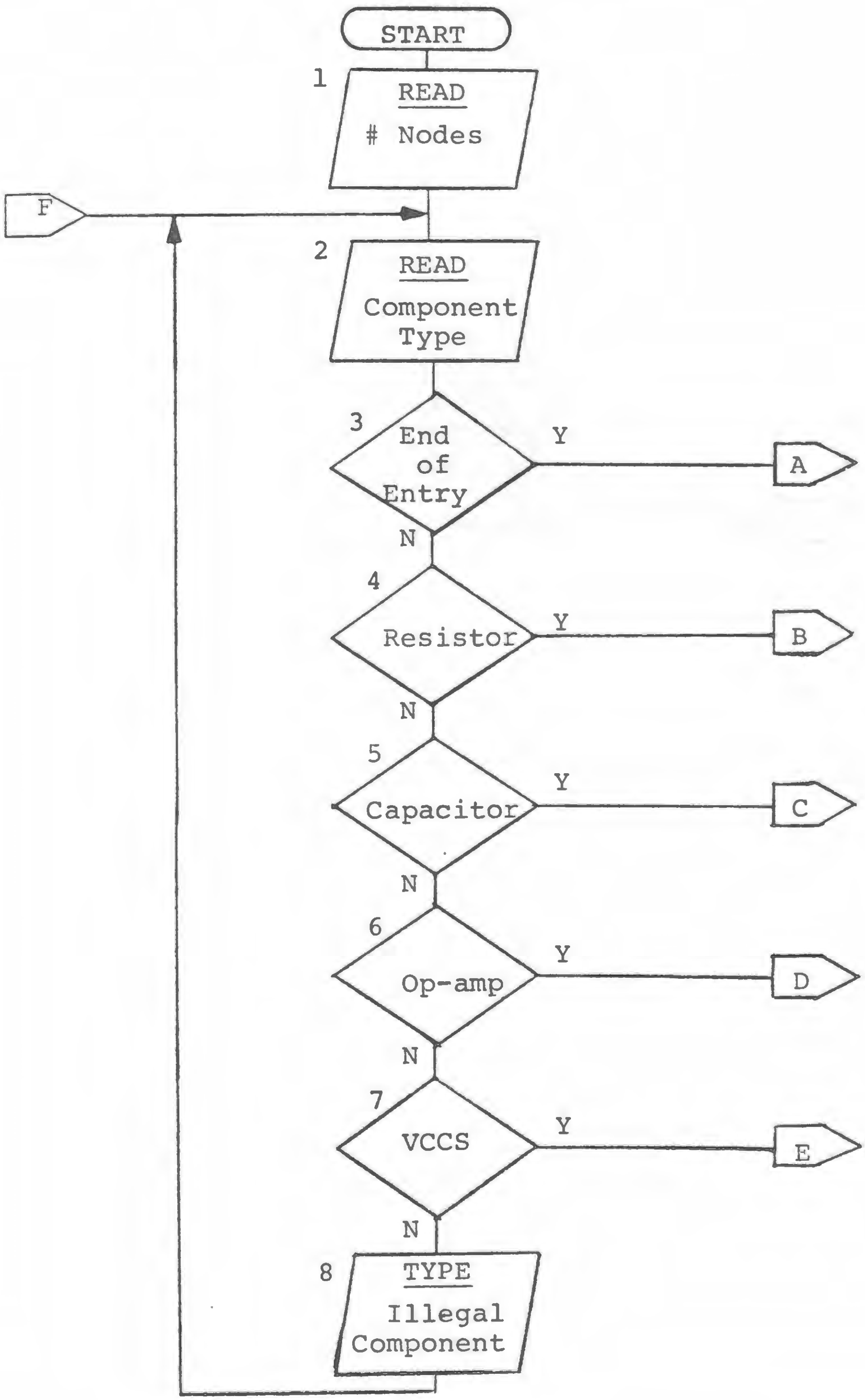
Input resistance	1M ohm
Gain	160,000 V/V
Output resistance	50 ohms

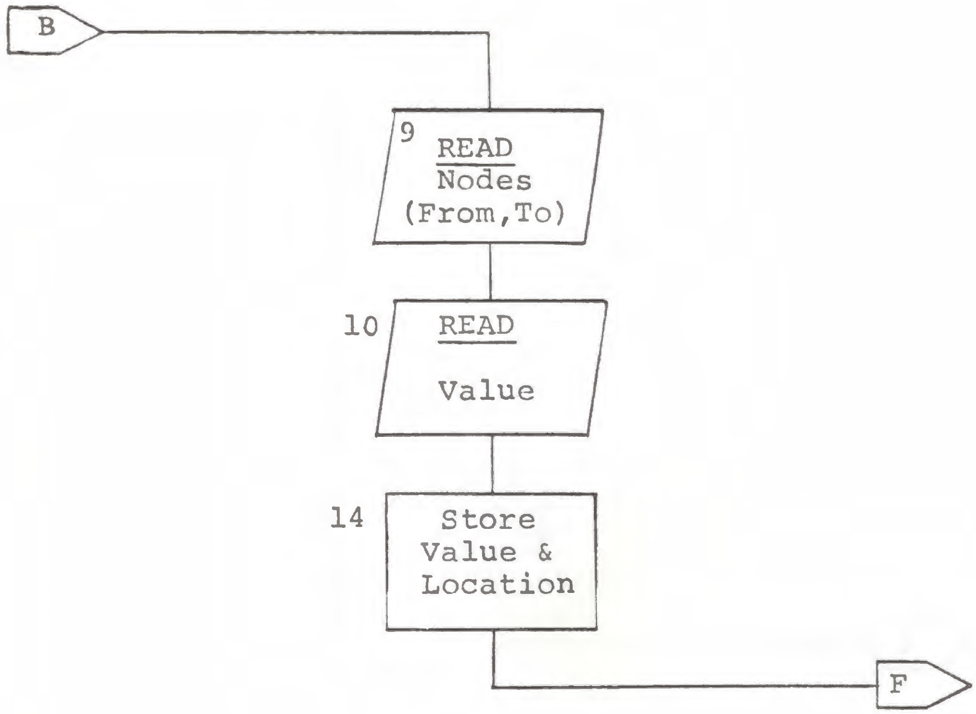
Operational Amplifier Characteristics

Table 4

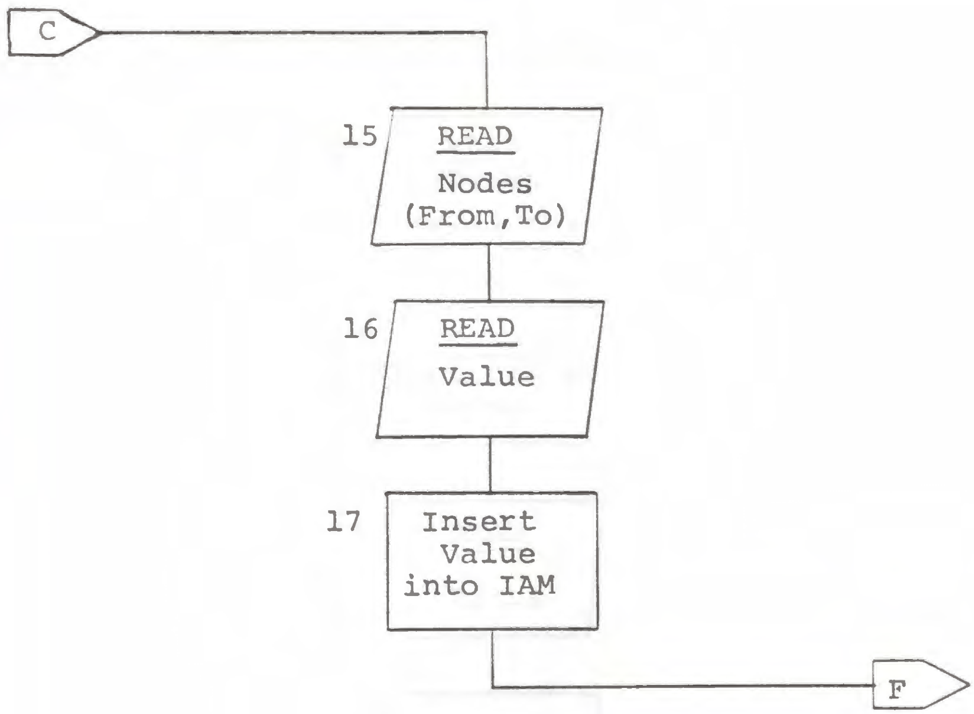
APPENDIX B

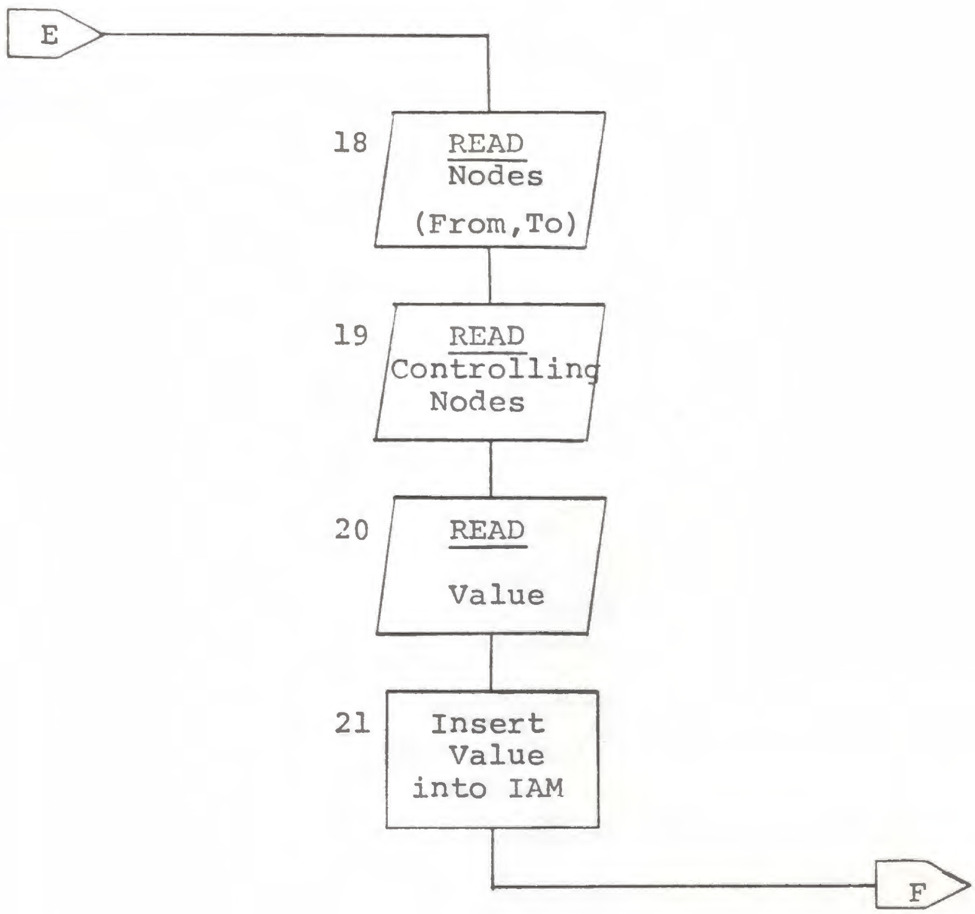
NOISE.F4 FLOW DIAGRAM

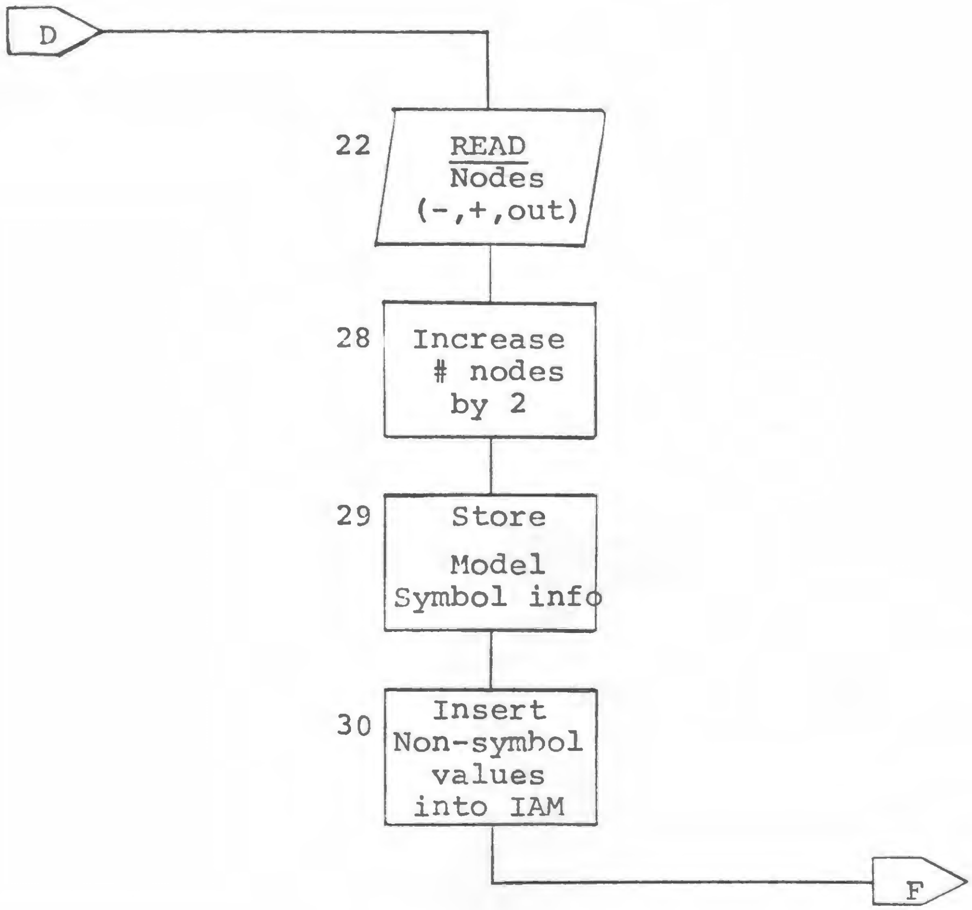




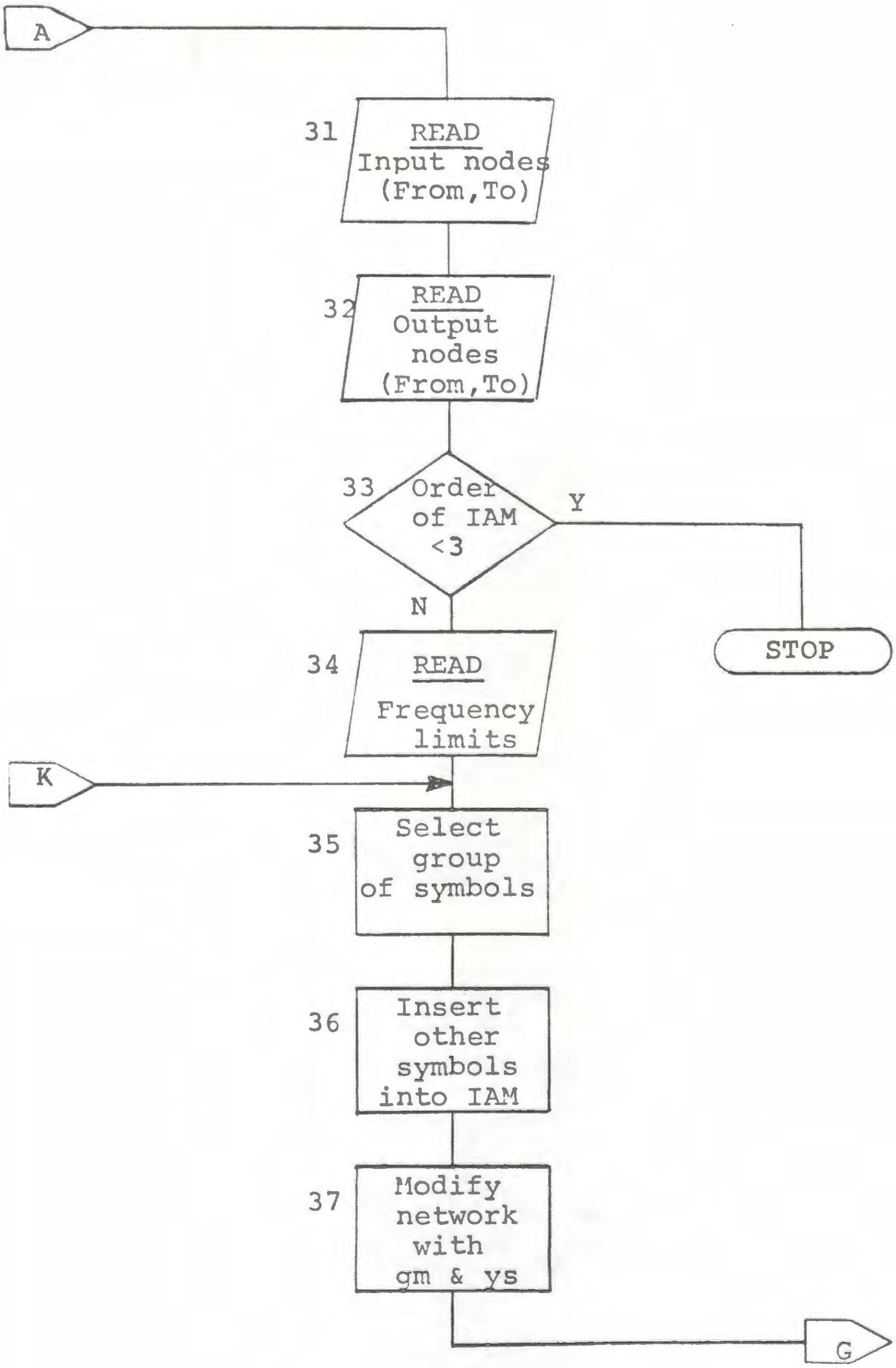
Blocks 11 thru 13 deleted

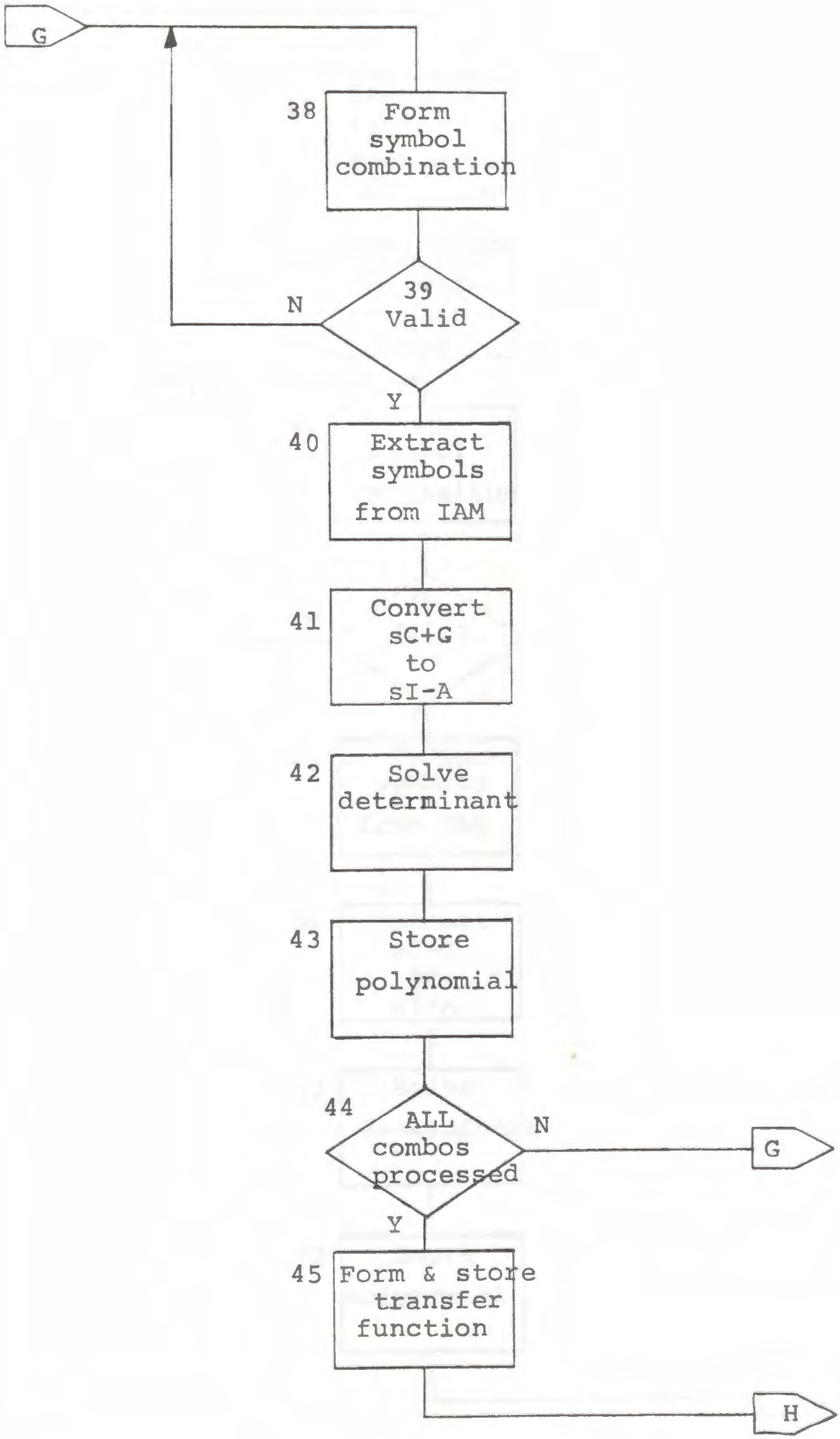

C
15
READ
Nodes
(From,To)
16
READ
Value
17
Insert
Value
into IAM
F

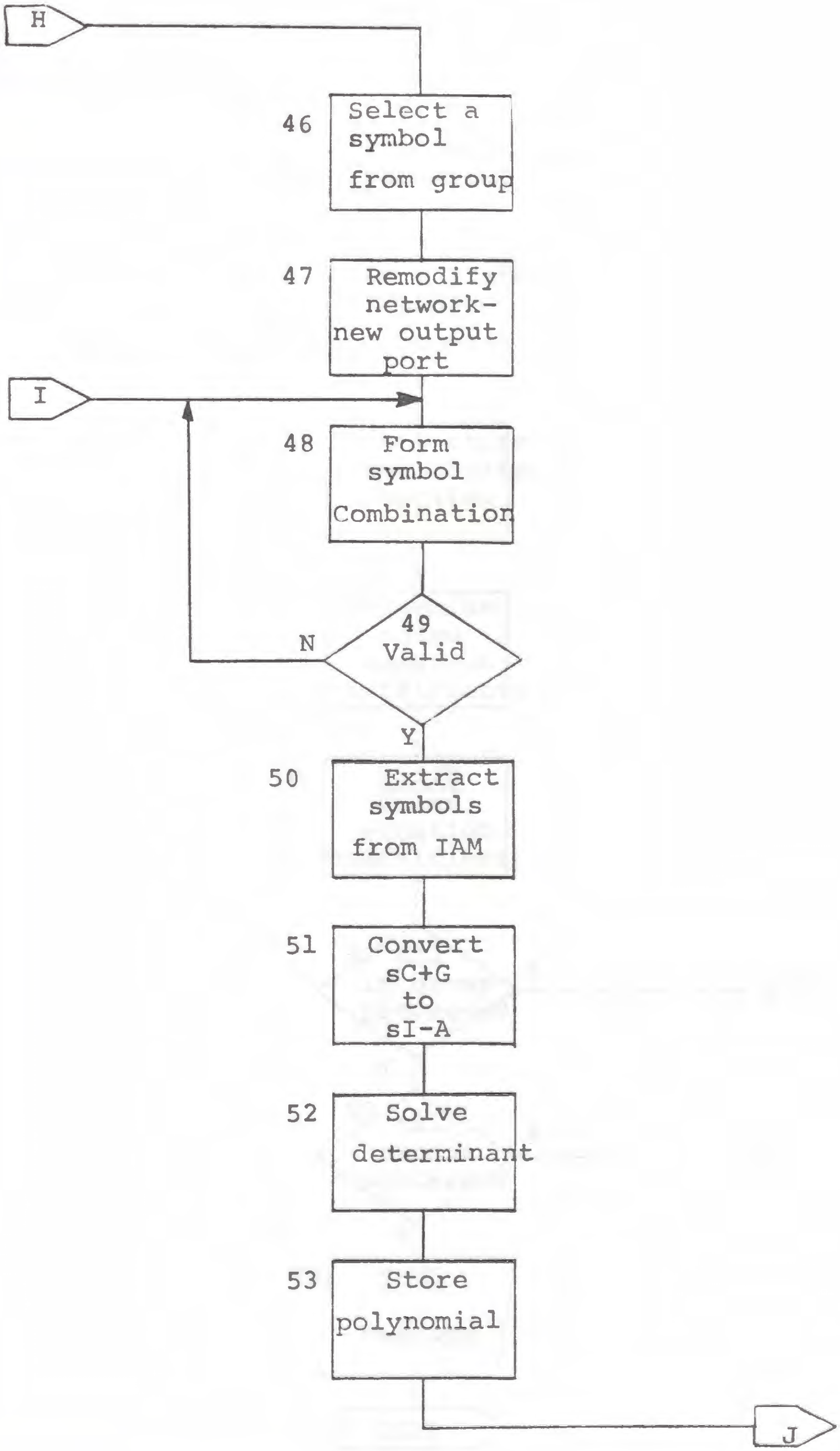

E
18
READ
Nodes
(From,To)
19
READ
Controlling
Nodes
20
READ
Value
21
Insert
Value
into IAM
F

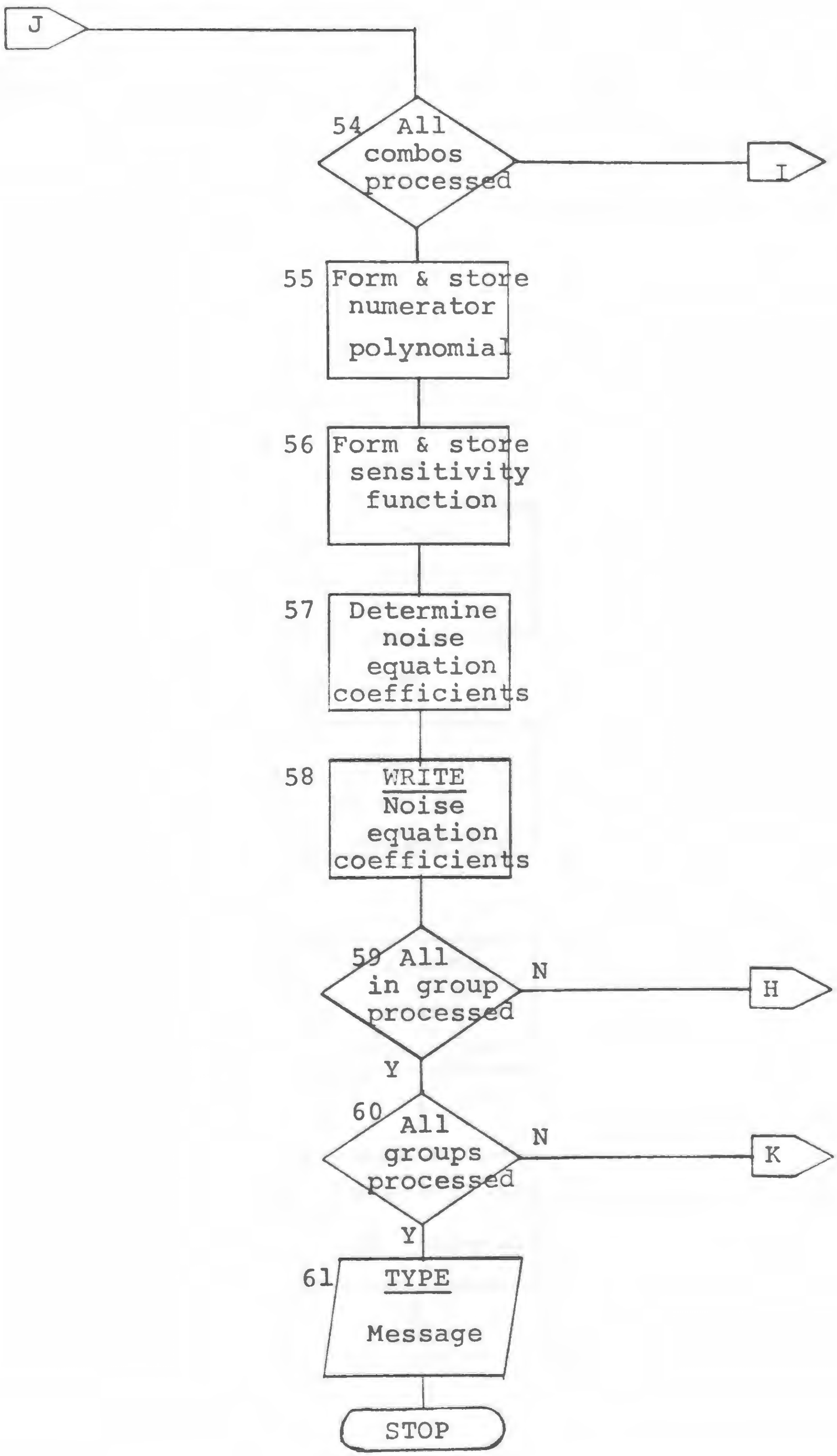


Blocks 23 thru 27 deleted


A
31
READ
Input nodes
(From,To)
32
READ
Output
nodes
(From,To)
33
Order
of IAM
<3
Y
STOP
N
34
READ
Frequency
limits
K
35
Select
group
of symbols
36
Insert
other
symbols
into IAM
37
Modify
network
with
gm & ys
G

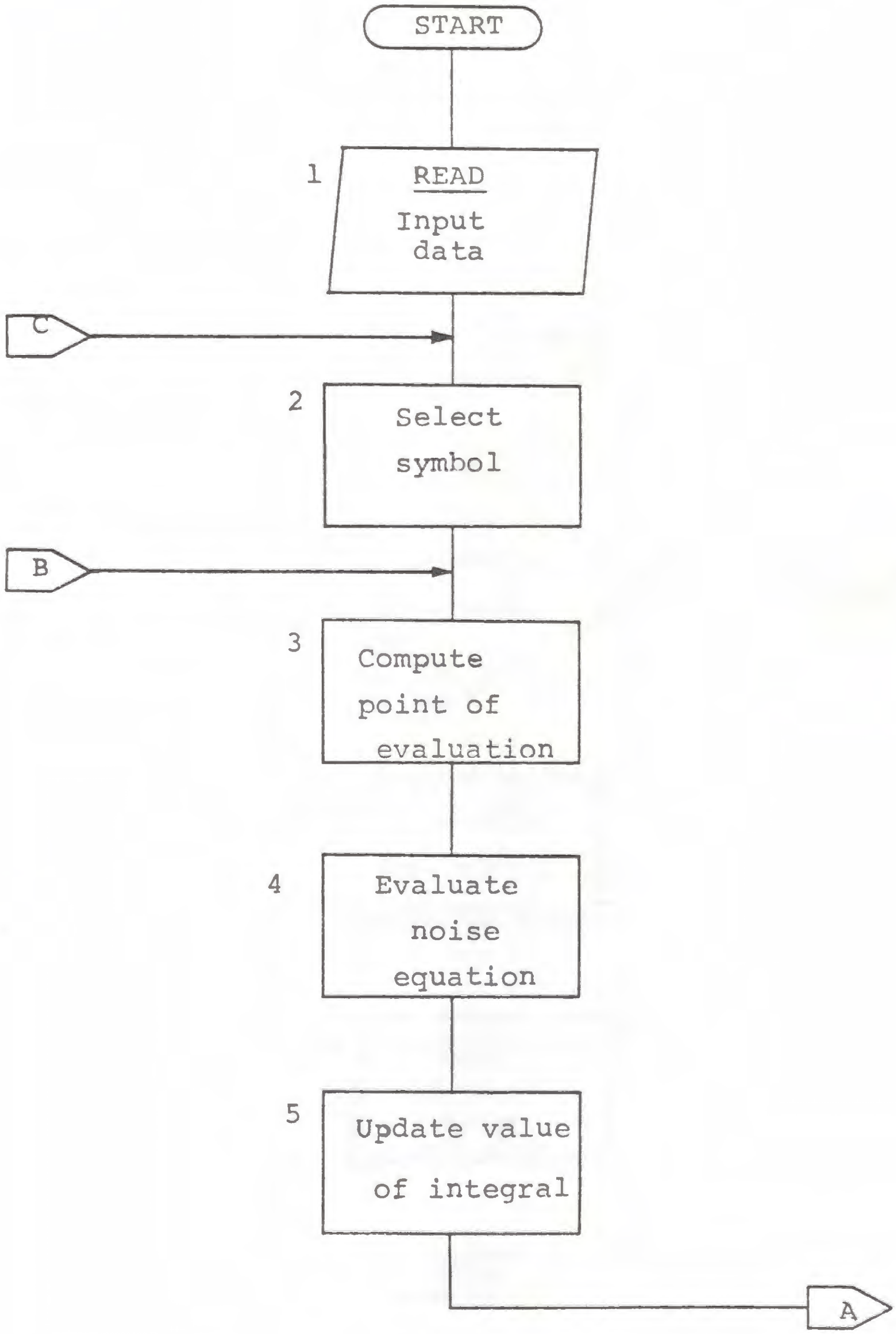

G
38 Form symbol combination
39 Valid
N
Y
40 Extract symbols from IAM
41 Convert sC+G to sI-A
42 Solve determinant
43 Store polynomial
44 ALL combos processed
N
G
Y
45 Form & store transfer function
H

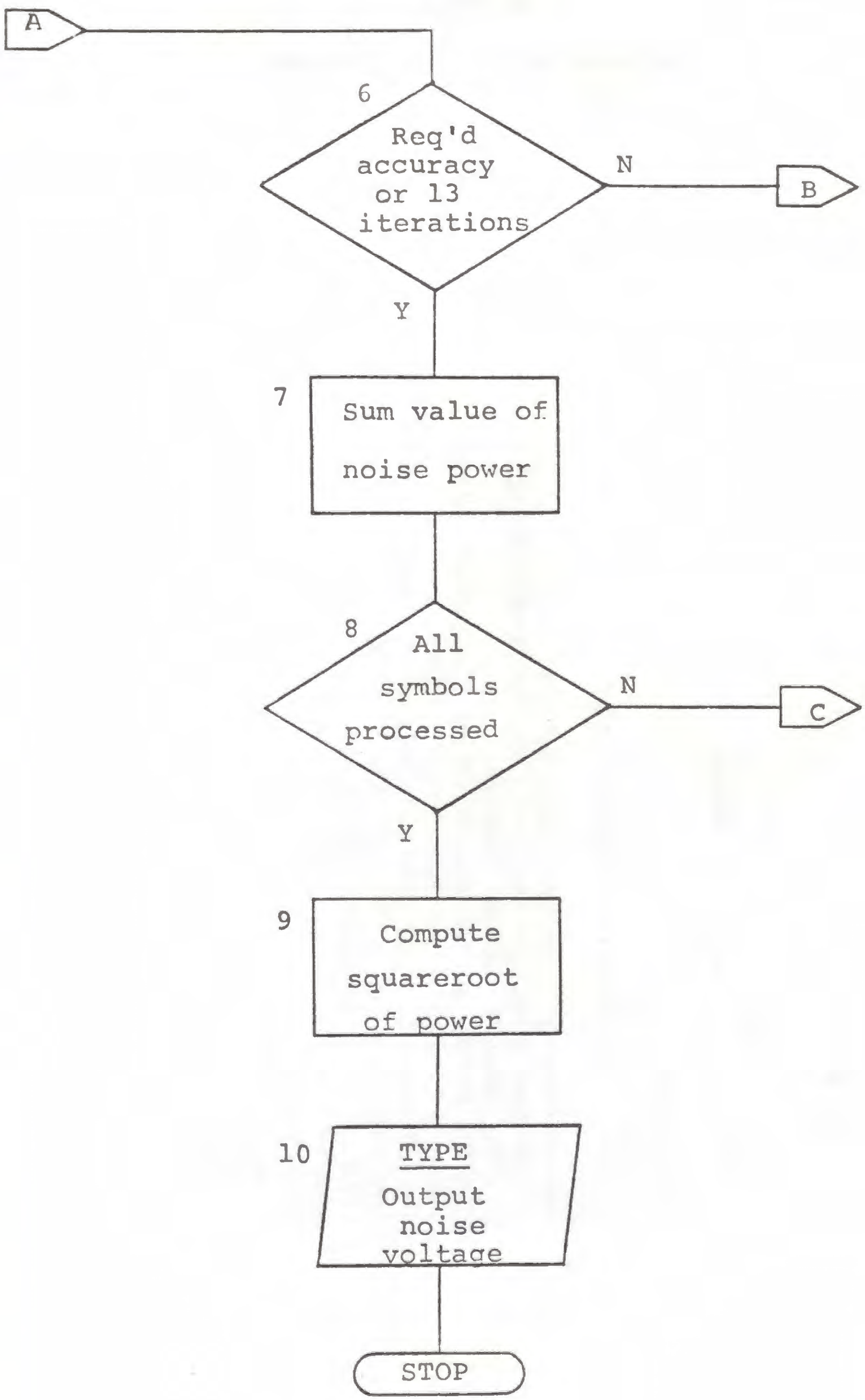

H
46 Select a symbol from group
47 Remodify network-new output port
I
48 Form symbol Combination
N
49 Valid
Y
50 Extract symbols from IAM
51 Convert sC+G to sI-A
52 Solve determinant
53 Store polynomial
J


J
54 All combos processed
I
55 Form & store numerator polynomial
56 Form & store sensitivity function
57 Determine noise equation coefficients
58 WRITE Noise equation coefficients
59 All in group processed
N
H
Y
60 All groups processed
N
K
Y
61 TYPE Message
STOP

APPENDIX C

SOLVE.F4 FLOW DIAGRAM




A
6
Req'd accuracy or 13 iterations
N
B
Y
7
Sum value of noise power
8
All symbols processed
N
C
Y
9
Compute squareroot of power
10
TYPE
Output noise voltage
STOP

APPENDIX D

NOISE.F4 PROGRAM LISTING

```
      EXTERNAL NEON
      DOUBLE PRECISION DNAME
      INTEGER*4 SBOL,POS(40,5),OUT(2),ULOL,PROFSA(40),
     1PROFSB(40),PTRA,PTRB,Y,C,D,SYMB,ROWI,ROWJ,POFSA2(40),
     2POFSB2(40)
      REAL*8 IAM1(20,20),IAM2(20,20),MIDVLT,MIDCUR,NOISE,
     1IPN(4),IPD(4),NEON,RPN(5),RPD(5),F(20,20),G(20,20),
     2VAL(40),XO(20),FLAT(20),COEFFA(40),COEFFB(40),
     3PYSOUT(5),AIJ(5),BKM(5),PGMOUT(5),COEFA2(40),COEFB2(40)
     4,PGMJ(5),FF(20,20),GF(20,20),SGN(14)
      REAL*8 VALUE,G1,G2,G0,GG,CG,G3,XULIM,XLLIM,RAN
      REAL*8 UNSCAL,FNORM1,FNORM2
      DIMENSION NODE(3),IN(2),L(20,20,5),MA(40,18),
     1MB(40,18),MA2(40,18),MB2(40,18)
      TYPE 1
    1 FORMAT('1','INPUT FILENAME=',$)
      ACCEPT 2,DNAME
    2 FORMAT(A10)
      TYPE 3
    3 FORMAT(' ','OUTPUT FILENAME=',$)
      ACCEPT 4, OFILE
    4 FORMAT(A5)
      DO 5 I=1,20
      FLAT(I)=0.
      XO(I)=0.
      DO 5 J=1,20
      F(I,J)=0.
      G(I,J)=0.
    5 CONTINUE
      DO 8 I=1,40
      VAL(I)=0.
```

```
        DO 8 J=1,5
        POS(I,J)=0
8       CONTINUE
        DO 9 J=1,3
        NODE(J)=0
9       CONTINUE
        SBOL=0
        OPEN(UNIT=1,FILE=DNAME)
        READ(1,20) N
20      FORMAT(1I)
30      READ(1,50) TYPE
50      FORMAT(A2)
        IF(TYPE.EQ.' ')GO TO 700
        IF(TYPE.EQ.'GM')GO TO 600
        IF(TYPE.EQ.'OA')GO TO 160
        READ(1,70) NODE(1),NODE(2)
70      FORMAT(2I)
        IF(NODE(2).GT.NODE(1))GO TO 75
        I=NODE(2)
        NODE(2)=NODE(1)
        NODE(1)=I
75      READ(1,90) VALUE
90      FORMAT(1F)
        IF(TYPE.EQ.'C')GO TO 150
        IF(TYPE.EQ.'R')GO TO 110
        TYPE 100
100     FORMAT(' ','*****INPUT ERROR----ILLEGAL COMPONENT*****')
        STOP
110     SBOL=SBOL+1
        POS(SBOL,1)=NODE(1)
        POS(SBOL,2)=NODE(2)
        POS(SBOL,3)=NODE(1)
        POS(SBOL,4)=NODE(2)
        POS(SBOL,5)=SBOL
        VAL(SBOL)=VALUE
        XO(SBOL)=0.
```

```
      FLAT(SBOL)=VALUE*2.635605858E-21
      GO TO 30
150   VALUE=VALUE*1E+9
      F(NODE(1),NODE(1))=F(NODE(1),NODE(1))+VALUE
      F(NODE(1),NODE(2))=F(NODE(1),NODE(2))-VALUE
      F(NODE(2),NODE(1))=F(NODE(2),NODE(1))-VALUE
      F(NODE(2),NODE(2))=F(NODE(2),NODE(2))+VALUE
      GO TO 30
160   READ(1,180) NODE(1),NODE(2),NODE(3)
180   FORMAT(3I)
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
/
      N=N+2
      SBOL=SBOL+1
/
/
/
```

```
/
/
/
      POS(SBOL,1)=NODE(1)
      POS(SBOL,2)=N-1
      POS(SBOL,3)=NODE(1)
      POS(SBOL,4)=N-1
      POS(SBOL,5)=SBOL
      FLAT(SBOL)=7.E-17
      XO(SBOL)=785.4
      VAL(SBOL)=10.
      IF(NODE(1).EQ.1)GO TO 290
      SBOL=SBOL+1
      POS(SBOL,1)=1
      POS(SBOL,2)=N-1
      POS(SBOL,3)=1
      POS(SBOL,4)=N-1
      POS(SBOL,5)=SBOL
      FLAT(SBOL)=2.39E-9
      XO(SBOL)=785.4
      VAL(SBOL)=1.E+8
290   IF(NODE(2).EQ.1)GO TO 300
      SBOL=SBOL+1
      POS(SBOL,1)=1
      POS(SBOL,2)=NODE(2)
      POS(SBOL,3)=1
      POS(SBOL,4)=NODE(2)
      POS(SBOL,5)=SBOL
      FLAT(SBOL)=2.39E-9
      XO(SBOL)=785.4
      VAL(SBOL)=1.E+8
300   G1=1.
      G2=.02
      G0=.02
      GG=6.25E-6
      G3=1E-6
```

```
      G(1,NODE(2))=G(1,NODE(2))+G1
      G(1,N-1)=G(1,N-1)-G1
      G(N,NODE(2))=G(N,NODE(2))-G1
      G(N,N-1)=G(N,N-1)+G1
      G(1,1)=G(1,1)+GG
      G(1,N)=G(1,N)-GG
      G(N,1)=G(N,1)-GG
      G(N,N)=G(N,N)+GG
      G(1,1)=G(1,1)+G0
      G(1,NODE(3))=G(1,NODE(3))-G0
      G(NODE(3),1)=G(NODE(3),1)-G0
      G(NODE(3),NODE(3))=G(NODE(3),NODE(3))+G0
      G(1,1)=G(1,1)-G2
      G(1,N)=G(1,N)+G2
      G(NODE(3),1)=G(NODE(3),1)+G2
      G(NODE(3),N)=G(NODE(3),N)-G2
      G(NODE(2),NODE(2))=G(NODE(2),NODE(2))+G3
      G(NODE(2),N-1)=G(NODE(2),N-1)-G3
      G(N-1,NODE(2))=G(N-1,NODE(2))-G3
      G(N-1,N-1)=G(N-1,N-1)+G3
      GO TO 30
/
/
/
/
/
/
/
/
/
/
/
600   READ(1,620) NODE(1),NODE(2)
620   FORMAT(2I)
      ROWI=NODE(1)
      ROWJ=NODE(2)
```

```
        READ(1,640) NODE(1),NODE(2)
640     FORMAT(2I)
        READ(1,660) VALUE
660     FORMAT(1F)
        G(ROWI,NODE(1))=G(ROWI,NODE(1))+VALUE
        G(ROWI,NODE(2))=G(ROWI,NODE(2))-VALUE
        G(ROWJ,NODE(1))=G(ROWJ,NODE(1))-VALUE
        G(ROWJ,NODE(2))=G(ROWJ,NODE(2))+VALUE
        GO TO 30
700     READ(1,720) IN(1),IN(2)
720     FORMAT(2I)
        READ(1,740) OUT(1),OUT(2)
740     FORMAT(2I)
        IF(IN(2).GT.IN(1))GO TO 743
        I=IN(2)
        IN(2)=IN(1)
        IN(1)=I
743     IF(OUT(2).GT.OUT(1))GO TO 745
        I=OUT(2)
        OUT(2)=OUT(1)
        OUT(1)=I
745     IF(N.LE.3)STOP
747     READ(1,758) XLLIM
        READ(1,758) XULIM
758     FORMAT(1F)
        CLOSE(UNIT=1,FILE=DNAME)
/
/
        OPEN(UNIT=1,FILE=OFILE)
        NOISE=0.
        J=N-3
        IF(J.GT.3)J=3
        IMAX=SBOL/J
        IF((IMAX*J).EQ.SBOL)IMAX=IMAX-1
        II=0
        JAY=J
```

```
770   ULOL=II*JAY
      LLOU=(II+1)*JAY+1
      DO 775 I=1,20
      DO 775 J=1,20
      IAM1(I,J)=F(I,J)
      IAM2(I,J)=G(I,J)
775   CONTINUE
      IF(ULOL.EQ.0)GO TO 790
      DO 780 NUM=1,ULOL
      IAM2(POS(NUM,1),POS(NUM,1))=IAM2(POS(NUM,1),POS(NUM,1))
      1+1/VAL(POS(NUM,5))
      IAM2(POS(NUM,1),POS(NUM,2))=IAM2(POS(NUM,1),POS(NUM,2))
      1-1/VAL(POS(NUM,5))
      IAM2(POS(NUM,2),POS(NUM,1))=IAM2(POS(NUM,2),POS(NUM,1))
      1-1/VAL(POS(NUM,5))
      IAM2(POS(NUM,2),POS(NUM,2))=IAM2(POS(NUM,2),POS(NUM,2))
      1+1/VAL(POS(NUM,5))
780   CONTINUE
790   NUM1=ULOL+1
      IF(II.EQ.IMAX)GO TO 810
      NUM2=LLOU-1
      DO 800 NUM=LLOU,SBOL
      IAM2(POS(NUM,1),POS(NUM,1))=IAM2(POS(NUM,1),POS(NUM,1))
      1+1/VAL(POS(NUM,5))
      IAM2(POS(NUM,1),POS(NUM,2))=IAM2(POS(NUM,1),POS(NUM,2))
      1-1/VAL(POS(NUM,5))
      IAM2(POS(NUM,2),POS(NUM,1))=IAM2(POS(NUM,2),POS(NUM,1))
      1-1/VAL(POS(NUM,5))
      IAM2(POS(NUM,2),POS(NUM,2))=IAM2(POS(NUM,2),POS(NUM,2))
      1+1/VAL(POS(NUM,5))
800   CONTINUE
      NUM=NUM2-NUM1+1
      GO TO 820
810   NUM=SBOL-NUM1+1
820   DO 825 JJ=1,20
      DO 825 KK=1,20
```

```
      DO 825 LL=1,5
      L(JJ,KK,LL)=0
825   CONTINUE
      DO 830 J=1,NUM
      DO 830 K=1,5
      L(J,1,K)=POS(J+NUM1-1,K)
830   CONTINUE
      L(NUM+1,1,1)=IN(1)
      L(NUM+1,1,2)=IN(2)
      L(NUM+1,1,3)=OUT(1)
      L(NUM+1,1,4)=OUT(2)
      L(NUM+1,1,5)=L(NUM,1,5)+1
      L(NUM+2,1,1)=IN(1)
      L(NUM+2,1,2)=IN(2)
      L(NUM+2,1,3)=IN(1)
      L(NUM+2,1,4)=IN(2)
      L(NUM+2,1,5)=L(NUM,1,5)+2
      SYMB=NUM+2
      DO 840 I=1,40
      DO 840 J=1,18
      MA(I,J)=0
      MB(I,J)=0
840   CONTINUE
      DO 850 I=1,40
      COEFFA(I)=0.
      COEFFB(I)=0.
      PROFSA(I)=-1
      PROFSB(I)=-1
850   CONTINUE
      C=0
      DO 860 I=1,5
      PGMOUT(I)=0.
      PYSOUT(I)=0.
860   CONTINUE
      DO 865 I=1,20
      DO 865 J=1,20
```

```
      FF(I,J)=IAM1(I,J)
      GF(I,J)=IAM2(I,J)
865   CONTINUE
      PTRA=0
      PTRB=0
      CALL PAREXT(FF,GF,N,SYMB,VAL,NUM,L,C,MA,MB,
     1COEFFA,COEFFB,PROFSA,PROFSB,PTRA,PTRB)
      NPTRA=PTRA
      NPTRB=PTRB
      DO 870 I=1,PTRA
      IF(PROFSA(I).LT.0)GO TO 870
      PGMOUT(PROFSA(I))=PGMOUT(PROFSA(I))+COEFFA(I)
870   CONTINUE
      DO 880 I=1,PTRB
      IF(PROFSB(I).LT.0)GO TO 880
      PYSOUT(PROFSB(I))=PYSOUT(PROFSB(I))+COEFFB(I)
880   CONTINUE
      DO 881 I=1,5
      IF(DABS(PYSOUT(I))-1.D-35)882,882,881
881   CONTINUE
882   I=I-1
      IF(I.EQ.0)GO TO 883
      FNORM1=1.D0/PYSOUT(I)
      GO TO 884
883   FNORM1=1.D0
884   DO 885 I=1,5
      PYSOUT(I)=PYSOUT(I)*FNORM1
      PGMOUT(I)=PGMOUT(I)*FNORM1
885   CONTINUE
      DO 886 I=2,5
      NPWRS=I-1
      UNSCAL=(1.D-9)**NPWRS
      PYSOUT(I)=PYSOUT(I)*UNSCAL
      PGMOUT(I)=PGMOUT(I)*UNSCAL
886   CONTINUE
      DO 890 I=1,5
```

```
        IF(DABS(PYSOUT(I))-1.D-35)892,892,890
890     CONTINUE
892     I=I-1
        IF(I.EQ.0)GO TO 893
        FNORM2=1.DO/PYSOUT(I)
        GO TO 894
893     FNORM2=1.DO
894     DO 895 I=1,5
        PYSOUT(I)=PYSOUT(I)*FNORM2
        PGMOUT(I)=PGMOUT(I)*FNORM2
895     CONTINUE
        IF(ULOL.NE.0)GO TO 904
        DO 6000 I=1,5
        TFN(I)=PGMOUT(6-I)
        TFD(I)=PYSOUT(6-I)
6000    CONTINUE
        TYPE 6002
6002    FORMAT('1')
        TYPE 896
896     FORMAT(' ',23X,'VOLTAGE TRANSFER FUNCTION')
        TYPE 897
897     FORMAT(' ',19X,'4',11X,'3',11X,'2')
        TYPE 898
898     FORMAT(' ',18X,'S',11X,'S',11X,'S',11X,'S')
        TYPE 900,(TFN(I),I=1,5)
900     FORMAT(' ','NUMERATOR =   ',4(D9.2,3X),D9.2)
        TYPE 902,(TFD(I),I=1,5)
902     FORMAT(' ','DENOMINATOR = ',4(D9.2,3X),D9.2)
904     CONTINUE
```

```
      DO 1050 Y=1,NUM
      L(NUM+1,1,3)=L(Y,1,3)
      L(NUM+1,1,4)=L(Y,1,4)
      DO 905 I=1,5
      FGMJ(I)=0.
      AIJ(I)=0.
      BKM(I)=0.
905   CONTINUE
      DO 908 I=1,40
      COEFA2(I)=0.
      COEFB2(I)=0.
      POFSA2(I)=-1
      POFSB2(I)=-1
908   CONTINUE
      DO 910 I=1,40
      DO 910 J=1,18
      MA2(I,J)=0
      MB2(I,J)=0
910   CONTINUE
      DO 920 I=1,20
      DO 920 J=1,20
      FF(I,J)=IAM1(I,J)
      GF(I,J)=IAM2(I,J)
920   CONTINUE
      C=1
```

```
      PTRA=0
      PTRB=0
      CALL PAREXT(FF,GF,N,SYMB,VAL,NUM,L,C,MA2,MB2,
     1COEFA2,COEFB2,POFSA2,POFSB2,PTRA,PTRB)
      DO 930 I=1,PTRA
      IF(POFSA2(I).LT.0)GO TO 930
      PGMJ(POFSA2(I))=PGMJ(POFSA2(I))+COEFA2(I)
930   CONTINUE
      DO 940 I=1,NPTRA
      IF(MA(I,L(Y,1,5)).EQ.1)GO TO 935
      GO TO 940
935   AIJ(PROFSA(I))=AIJ(PROFSA(I))+COEFFA(I)
940   CONTINUE
      DO 950 I=1,NPTRB
      IF(MB(I,L(Y,1,5)).EQ.1)GO TO 945
      GO TO 950
945   BKM(PROFSB(I))=BKM(PROFSB(I))+COEFFB(I)
950   CONTINUE
      DO 955 I=1,5
      PGMJ(I)=PGMJ(I)*FNORM1
      AIJ(I)=AIJ(I)*FNORM1
      BKM(I)=BKM(I)*FNORM1
955   CONTINUE
      DO 957 I=2,5
      NPWRS=I-1
      UNSCAL=(1.D-9)**NPWRS
      PGMJ(I)=PGMJ(I)*UNSCAL
      AIJ(I)=AIJ(I)*UNSCAL
      BKM(I)=BKM(I)*UNSCAL
957   CONTINUE
      DO 958 I=1,5
      PGMJ(I)=PGMJ(I)*FNORM2
      AIJ(I)=AIJ(I)*FNORM2
      BKM(I)=BKM(I)*FNORM2
958   CONTINUE
      TYPE 2000,(AIJ(I),I=1,5)
```

```
      TYPE 2001,(BKM(I),I=1,5)
      TYPE 2002,(PYSOUT(I),I=1,5)
      TYPE 2003,(PGMOUT(I),I=1,5)
      TYPE 2004,(PGMJ(I),I=1,5)
      TYPE 2005,FLAT(Y+ULOL)
      TYPE 2006,XO(Y+ULOL)
2000  FORMAT(' ','    AIJ=',5(3X,E10.3))
2001  FORMAT(' ','    BKM=',5(3X,E10.3))
2002  FORMAT(' ','PYSOUT=',5(3X,E10.3))
2003  FORMAT(' ','PGMOUT=',5(3X,E10.3))
2004  FORMAT(' ','   PGMJ=',5(3x,E10.3))
2005  FORMAT(' ','FLAT=',3X,E10.3)
2006  FORMAT(' ','XO   =',3X,E10.3)
1000  CONTINUE
      RPN(1)=AIJ(1)*PYSOUT(1)-PGMOUT(1)*BKM(1)
      RPN(2)=PGMOUT(3)*BKM(1)+PGMOUT(1)*BKM(3)+PGMOUT
     1(2)*BKM(2)-AIJ(3)*PYSOUT(1)-AIJ(1)*PYSOUT(3)-AIJ(2)
     2*PYSOUT(2)
      RPN(3)=AIJ(5)*PYSOUT(1)+AIJ(3)*PYSOUT(3)+AIJ(1)*
     1PYSOUT(5)+AIJ(2)*PYSOUT(4)+AIJ(4)*PYSOUT(2)-PGMOUT
     2(5)*BKM(1)-PGMOUT(3)*BKM(3)-PGMOUT(1)*BKM(5)-
     3PGMOUT(2)*BKM(4)-PGMOUT(4)*BKM(2)
      RPN(4)=PGMOUT(5)*BKM(3)+PGMOUT(3)*BKM(5)+PGMOUT(4)*
     1BKM(4)-AIJ(5)*PYSOUT(3)-AIJ(3)*PYSOUT(5)-AIJ(4)*
     2PYSOUT(4)
      RPN(5)=AIJ(5)*PYSOUT(5)-PGMOUT(5)*BKM(5)
      IPN(1)=AIJ(2)*PYSOUT(1)+AIJ(1)*PYSOUT(2)-PGMOUT(2)*BKM
     1(1)-PGMOUT(1)*BKM(2)
      IPN(2)=PGMOUT(2)*BKM(3)+PGMOUT(4)*BKM(1)+PGMOUT(1)*BKM
     1(4)+PGMOUT(3)*BKM(2)-AIJ(2)*PYSOUT(3)-AIJ(4)*PYSOUT(1)
     2-AIJ(1)*PYSOUT(4)-AIJ(3)*PYSOUT(2)
      IPN(3)=AIJ(2)*PYSOUT(5)+AIJ(4)*PYSOUT(3)+AIJ(3)*PYSOUT
     1(4)+AIJ(5)*PYSOUT(2)-PGMOUT(2)*BKM(5)-PGMOUT(4)*BKM(3)
     2-PGMOUT(3)*BKM(4)-PGMOUT(5)*BKM(2)
      IPN(4)=PGMOUT(4)*BKM(5)+PGMOUT(5)*BKM(4)-AIJ(4)*PYSOUT
     1(5)-AIJ(5)*PYSOUT(4)
```

```
RPD(1)=PGMJ(1)*PYSOUT(1)
RPD(2)=-PGMJ(3)*PYSOUT(1)-PGMJ(1)*PYSOUT(3)-PGMJ(2)*
1PYSOUT(2)
RPD(3)=PGMJ(5)*PYSOUT(1)+PGMJ(3)*PYSOUT(3)+PGMJ(1)*
1PYSOUT(5)+PGMJ(2)*PYSOUT(4)+PGMJ(4)*PYSOUT(2)
RPD(4)=-PGMJ(5)*PYSOUT(3)-PGMJ(3)*PYSOUT(5)-PGMJ(4)*
1PYSOUT(4)
RPD(5)=PGMJ(5)*PYSOUT(5)
IPD(1)=PGMJ(2)*PYSOUT(1)+PGMJ(1)*PYSOUT(2)
IPD(2)=-PGMJ(2)*PYSOUT(3)-PGMJ(4)*PYSOUT(1)-PGMJ(1)*
1PYSOUT(4)-PGMJ(3)*PYSOUT(2)
IPD(3)=PGMJ(2)*PYSOUT(5)+PGMJ(4)*PYSOUT(3)+PGMJ(3)*
1PYSOUT(4)+PGMJ(5)*PYSOUT(2)
IPD(4)=-PGMJ(4)*PYSOUT(5)-PGMJ(5)*PYSOUT(4)
CALL SGNCHK(RPN(2),SGN(1))
RPN(2)=RPN(2)**.5
CALL SGNCHK(RPN(3),SGN(2))
RPN(3)=RPN(3)**.25
CALL SGNCHK(RPN(4),SGN(3))
RPN(4)=RPN(4)**(1./6.)
CALL SGNCHK(RPN(5),SGN(4))
RPN(5)=RPN(5)**.125
CALL SGNCHK(IPN(2),SGN(5))
IPN(2)=IPN(2)**(1./3.)
CALL SGNCHK(IPN(3),SGN(6))
IPN(3)=IPN(3)**.2
CALL SGNCHK(IPN(4),SGN(7))
IPN(4)=IPN(4)**(1./7.)
CALL SGNCHK(RPD(2),SGN(8))
RPD(2)=RPD(2)**.5
CALL SGNCHK(RPD(3),SGN(9))
RPD(3)=RPD(3)**.25
CALL SGNCHK(RPD(4),SGN(10))
RPD(4)=RPD(4)**(1./6.)
CALL SGNCHK(RPD(5),SGN(11))
RPD(5)=RPD(5)**.125
```

```
      CALL SGNCHK(IPD(2),SGN(12))
      IPD(2)=IPD(2)**(1./3.)
      CALL SGNCHK(IPD(3),SGN(13))
      IPD(3)=IPD(3)**.2
      CALL SGNCHK(IPD(4),SGN(14))
      IPD(4)=IPD(4)**(1./7.)
      WRITE(1,3000),(RPN(JJJ),JJJ=1,5)
      WRITE(1,3000),(IPN(JJJ),JJJ=1,4)
      WRITE(1,3000),(RPD(JJJ),JJJ=1,5)
      WRITE(1,3000),(IPD(JJJ),JJJ=1,4)
3000  FORMAT(5D10.3)
      LSUBY=Y
      XL=XLLIM*6.28318
      XU=XULIM*6.28318
      WRITE(1,4000),(FLAT(JJJ),JJJ=1,20)
      WRITE(1,4000),(XO(JJJ),JJJ=1,20)
      WRITE(1,4010),(SGN(JJJ),JJJ=1,14)
      WRITE(1,4020),XL,XU,LSUBY,ULOL
4000  FORMAT(20D10.3)
4010  FORMAT(14D8.1)
4020  FORMAT(2D10.3,2I2)
1050  CONTINUE
      II=II+1
      IF(II.GT.IMAX)GO TO 1060
      GO TO 770
1060  CLOSE(UNIT=1,FILE=OFILE)
      TYPE 4030,OFILE
4030  FORMAT(' ','+++EXECUTE PROGRAM CALLED SOLVE AND USE ',A5,' AS THE INPUT
     1FILENAME+++')
      STOP
      END
      SUBROUTINE PAREXT(FF,GF,N,SYMB,VAL,NUM,L,C,MA,MB,
     1COEFFA,COEFFB,PROFSA,PROFSB,PTRA,PTRB)
      INTEGER*4 SYMB,PROFSA,PROFSB,C,SSTRNG,MA,MB,P,S,T,TT,
     1T1,AA,BB,PTRA,PTRB,TEMP
      REAL*8 IAM1(20,20),IAM2(20,20),POLY(10),COEFFA(40),
```

```
      1COEFFB(40),F(20,20),G(20,20),VAL(40),FF(20,20),
      2GF(20,20),DETFGI
      DIMENSION SSTRNG(10),MA(40,18),MB(40,18),L(20,20,5),
      1P(20),S(20),AA(5),BB(5),PROFSA(40),PROFSB(40),
      2MAP(40,18)
      GO TO 4061
      TYPE 7,N,SYMB,NUM
7     FORMAT(' ','N=',I2,3X,'SYMB=',I2,3X,'NUM=',I2)
      DO 8 II=1,SYMB
      TYPE 9,(L(II,1,KK),KK=1,5)
8     CONTINUE
9     FORMAT(' ',5(I1,1X))
4061  CONTINUE
      FINCH=1.D-12
      DO 10 M=1,20
      P(M)=0
      S(M)=0
10    CONTINUE
      M=1
      S(1)=SYMB
      T=N
20    P(M)=1
      I=P(M)
30    J=I+1
40    DO 50 K=1,5
      AA(K)=L(I,M,K)
      BB(K)=L(J,M,K)
50    CONTINUE
      IF(AA(2)-BB(2))60,70,80
60    BB(2)=BB(2)-1
      GO TO 80
70    BB(2)=AA(1)
80    IF(AA(2)-BB(1))90,100,110
90    BB(1)=BB(1)-1
      GO TO 110
100   BB(1)=AA(1)
```

```
110     IF(AA(4)-BB(4))120,130,140
120     BB(4)=BB(4)-1
        GO TO 140
130     BB(4)=AA(3)
140     IF(AA(4)-BB(3))150,160,170
150     BB(3)=BB(3)-1
        GO TO 170
160     BB(3)=AA(3)
170     IF(BB(1)-BB(2))200,230,190
190     TEMP=BB(1)
        BB(1)=BB(2)
        BB(2)=TEMP
200     IF(BB(3)-BB(4))210,230,205
205     TEMP=BB(3)
        BB(3)=BB(4)
        BB(4)=TEMP
210     S(M+1)=S(M+1)+1
        DO 220 K=1,5
        L(S(M+1),M+1,K)=BB(K)
220     CONTINUE
230     IF(J-S(M))240,250,240
240     J=J+1
        GO TO 40
250     M=M+1
        T=T-1
        IF(S(M))260,280,260
260     IF(S(M)-1)20,300,20
280     M=M-1
        T=T+1
        P(M)=P(M)+1
        I=P(M)
        IF(P(M)-S(M))30,310,30
300     P(M)=1
310     IF(C.EQ.1)GO TO 320
        IF(L(P(M),M,5)-L(NUM+2,1,5))320,350,320
320     IF(L(P(M),M,5)-L(NUM+1,1,5))370,350,370
```

```
330     S(M)=0
        P(M)=0
        IF(M.EQ.1)GO TO 540
        DO 340 NR=1,20
        DO 335 NC=1,5
        L(NR,M,NC)=0
335     CONTINUE
340     CONTINUE
        GO TO 280
350     DO 360 NB=1,N
        DO 360 NA=1,N
        IAM1(NB,NA)=FF(NB,NA)
        IAM2(NB,NA)=GF(NB,NA)
360     CONTINUE
        DETPQI=1.
        CALL MANIAM(IAM1,IAM2,N,DETPQI,L,M,P,TT,SSTRNG)
369     CONTINUE
        T1=TT-1
        IF(T1-1)370,380,380
370     IF(P(M).EQ.1)GO TO 330
        P(M)=P(M)-1
        GO TO 310
380     IF(C.EQ.1)GO TO 390
        IF(SSTRNG(M).EQ.L(NUM+2,1,5))GO TO 440
390     IF(SSTRNG(M).EQ.L(NUM+1,1,5))GO TO 450
        NO1=NUM+1
        NO2=NUM+2
        TYPE 400, M,NO1,NO2
400     FORMAT(' ','---ERROR--- SSTRNG(',I2,') IS NOT THE
        1SAME AS L(',I2,',1,5) OR L(',I2,',1,5)')
        STOP
440     CONTINUE
        CALL MATCON(IAM1,IAM2,T1,DETPQI,POLY,NCOEFF)
        CALL FFORM(DETPQI,POLY,NCOEFF,SSTRNG,M,PTRB,VAL,
        1MB,COEFFB,PROFSB)
        GO TO 370
```

```
450     CONTINUE
        CALL MATCON(IAM1,IAM2,T1,DETPQI,POLY,NCOEFF)
        IF(C.EQ.1)GO TO 460
        GO TO 453
        DO 452 II=1,10
        TYPE 600,(MA(II,JJ),JJ=1,5)
452     CONTINUE
453     CONTINUE
        CALL FFORM(DETPQI,POLY,NCOEFF,SSTRNG,M,PTRA,VAL,
        1MA,COEFFA,PROFSA)
        GO TO 458
        DO 457 II=1,10
        TYPE 600,(MA(II,JJ),JJ=1,5)
457     CONTINUE
        TYPE 610,PTRA,(PROFSA(II),II=1,10)
        TYPE 620,(COEFFA(II),II=1,7)
458     CONTINUE
        GO TO 370
460     CALL FFORM(DETPQI,POLY,NCOEFF,SSTRNG,M,PTRA,VAL,
        1MAP,COEFFA,PROFSA)
        GO TO 370
540     RETURN
600     FORMAT(' ',4(I1,2X),I1)
610     FORMAT(' ','PTRA=  ',I2,'PROFSA=  ',9(I1,2X),I1)
620     FORMAT(' ',7E10.3)
        END
        SUBROUTINE MANIAM(IAM1,IAM2,N,DETPQI,L,M,P,TT,SSTRNG)
        REAL*8 IAM1(20,20),IAM2(20,20),DETPQI
        INTEGER*4 P,TT,V,SSTRNG
        DIMENSION P(20),L(20,20,5),SSTRNG(10)
        DO 6 K=1,10
        SSTRNG(K)=0
6       CONTINUE
        K=1
        TT=N
10      SSTRNG(K)=L(P(K),K,5)
```

```
      DO 20 KK=1,TT
      IAM1(L(P(K),K,1),KK)=IAM1(L(P(K),K,1),KK)+IAM1(L(P(K),
     1K,2),KK)
      IAM2(L(P(K),K,1),KK)=IAM2(L(P(K),K,1),KK)+IAM2(L(P(K),
     1K,2),KK)
20    CONTINUE
      DO 30 KK=1,TT
      IAM1(KK,L(P(K),K,3))=IAM1(KK,L(P(K),K,3))+IAM1(KK,
     1L(P(K),K,4))
      IAM2(KK,L(P(K),K,3))=IAM2(KK,L(P(K),K,3))+IAM2(KK,
     1L(P(K),K,4))
30    CONTINUE
      IF(L(P(K),K,2).EQ.TT)GO TO 45
      V=L(P(K),K,2)+1
      DO 40 KK=V,TT
      DO 40 J=1,TT
      IAM1(KK-1,J)=IAM1(KK,J)
      IAM2(KK-1,J)=IAM2(KK,J)
40    CONTINUE
45    IF(L(P(K),K,4).EQ.TT)GO TO 55
      V=L(P(K),K,4)+1
      DO 50 KK=V,TT
      DO 50 J=1,TT
      IAM1(J,KK-1)=IAM1(J,KK)
      IAM2(J,KK-1)=IAM2(J,KK)
50    CONTINUE
55    TT=TT-1
      I=L(P(K),K,2)+L(P(K),K,4)
      II=I
      I=(I/2)*2
      IF((II-I).EQ.1)DETPQI=-DETPQI
      IF(K.EQ.M)GO TO 100
      K=K+1
      GO TO 10
100   RETURN
200   FORMAT(' ','MANIAM')
```

```
      END
      SUBROUTINE MATCON(IAM1,IAM2,NN,DETPQI,POLY,NCOEFF)
      INTEGER*4 DIM,DIM3,SROW,SCOL,BB,QQ,RR,SS,WW,
     1VV,YY,ZZ,DD,EE
      REAL*8 IAM1(20,20),IAM2(20,20),POLY(10),DETPQI,PINCH,
     1ACHK,PCHK,RCHK,R
      PINCH=1.D-12
      K=0
      J=0
      BB=1
      DIM3=NN
7     BB=BB+K
      DIM3=DIM3-(K+J)
      DIM=DIM3-BB+1
      I=0
      J=0
      II=BB
10    IF(DABS(IAM1(II,II))-PINCH)40,40,20
20    IF(II.EQ.DIM3)GO TO 250
30    CALL SUB4(IAM1,IAM2,BB,DIM3,II,II,DIM3,DIM3)
      II=II+1
      GO TO 10
40    IAM1(II,II)=0.
      IF(II.EQ.DIM3)GO TO 240
      JJ=II+1
50    IF(DABS(IAM1(JJ,II))-PINCH)70,70,60
60    CALL SUB1(IAM1,IAM2,II,JJ,DETPQI,BB,DIM3)
      GO TO 30
70    IAM1(JJ,II)=0.
      JJ=JJ+1
      IF(JJ.GT.DIM3)GO TO 80
      GO TO 50
80    JJ=II+1
90    IF(DABS(IAM1(II,JJ))-PINCH)110,110,100
100   CALL SUB2(IAM1,IAM2,II,JJ,DETPQI,BB,DIM3)
      GO TO 30
```

```
110     IAM1(II,JJ)=0.
        JJ=JJ+1
        IF(JJ.GT.DIM3)GO TO 120
        GO TO 90
120     RR=II+1
130     IF(DABS(IAM1(RR,RR))-PINCH)150,150,140
140     CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
        CALL SUB2(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
        GO TO 30
150     IAM1(RR,RR)=0.
        IF(RR.EQ.DIM3)GO TO 240
        JJ=RR+1
160     IF(DABS(IAM1(JJ,RR))-PINCH)180,180,170
170     CALL SUB1(IAM1,IAM2,II,JJ,DETPQI,BB,DIM3)
        CALL SUB2(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
        GO TO 30
180     IAM1(JJ,RR)=0.
        JJ=JJ+1
        IF(JJ.GT.DIM3)GO TO 190
        GO TO 160
190     JJ=RR+1
200     IF(DABS(IAM1(RR,JJ))-PINCH)220,220,210
210     CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
        CALL SUB2(IAM1,IAM2,II,JJ,DETPQI,BB,DIM3)
        GO TO 30
220     IAM1(RR,JJ)=0.
        JJ=JJ+1
        IF(JJ.GT.DIM3)GO TO 230
        GO TO 200
230     RR=RR+1
        GO TO 130
240     I=II-BB
        LROW=II-1
        LCOL=II-1
        GO TO 252
250     I=II-BB+1
```

```
      LROW=II
      LCOL=II
252   IF(I.EQ.0)GO TO 253
      CALL SUB3(IAM1,IAM2,DETPQI,BB,DIM3,BB,BB,LROW,LCOL)
      IF(I.EQ.DIM)GO TO 257
      GO TO 258
253   CALL NOCAP(IAM1,IAM2,BB,DIM3,POLY,NCOEFF,DETPQI)
      GO TO 1060
257   CALL DET(IAM2,BB,I,POLY,NCOEFF,DETPQI)
      GO TO 1060
258   WW=BB+I
260   IF(DABS(IAM2(WW,WW))-PINCH)290,290,270
270   IF(WW.EQ.DIM3)GO TO 500
280   CALL SUB4(IAM2,IAM1,BB,DIM3,WW,WW,DIM3,DIM3)
      WW=WW+1
      GO TO 260
290   IAM2(WW,WW)=0.
      IF(WW.EQ.DIM3)GO TO 490
      QQ=WW+1
300   IF(DABS(IAM2(QQ,WW))-PINCH)320,320,310
310   CALL SUB1(IAM1,IAM2,WW,QQ,DETPQI,BB,DIM3)
      GO TO 280
320   IAM2(QQ,WW)=0.
      QQ=QQ+1
      IF(QQ.GT.DIM3)GO TO 330
      GO TO 300
330   QQ=WW+1
340   IF(DABS(IAM2(WW,QQ))-PINCH)360,360,350
350   CALL SUB2(IAM1,IAM2,WW,QQ,DETPQI,BB,DIM3)
      GO TO 280
360   IAM2(WW,QQ)=0.
      QQ=QQ+1
      IF(QQ.GT.DIM3)GO TO 370
      GO TO 340
370   RR=WW+1
380   IF(DABS(IAM2(RR,RR))-PINCH)400,400,390
```

```
390    CALL SUB1(IAM1,IAM2,WW,RR,DETPQI,BB,DIM3)
       CALL SUB2(IAM1,IAM2,WW,RR,DETPQI,BB,DIM3)
       GO TO 280
400    IAM2(RR,RR)=0.
       IF(RR.EQ.DIM3)GO TO 490
       JJ=RR+1
410    IF(DABS(IAM2(JJ,RR))-PINCH)430,430,420
420    CALL SUB1(IAM1,IAM2,WW,JJ,DETPQI,BB,DIM3)
       CALL SUB2(IAM1,IAM2,WW,RR,DETPQI,BB,DIM3)
       GO TO 280
430    IAM2(JJ,RR)=0.
       JJ=JJ+1
       IF(JJ.GT.DIM3)GO TO 440
       GO TO 410
440    JJ=RR+1
450    IF(DABS(IAM2(RR,JJ))-PINCH)470,470,460
460    CALL SUB1(IAM1,IAM2,WW,RR,DETPQI,BB,DIM3)
       CALL SUB2(IAM1,IAM2,WW,JJ,DETPQI,BB,DIM3)
       GO TO 280
470    IAM2(RR,JJ)=0.
       JJ=JJ+1
       IF(JJ.GT.DIM3)GO TO 480
       GO TO 450
480    RR=RR+1
       GO TO 380
490    J=WW-BB-I
       LROW=WW-1
       LCOL=WW-1
       GO TO 510
500    J=DIM-I
       LROW=DIM3
       LCOL=DIM3
510    ISROW=BB+I
       CALL SUB3(IAM2,IAM1,DETPQI,BB,DIM3,ISROW,ISROW,
      1LROW,LCOL)
       K=DIM-(I+J)
```

```
      IF(K.GT.I)GO TO 1080
      IF(J.EQ.0)GO TO 575
      MSTRT=BB+I
      MEND=BB+I+J-1
      DO 540 M=MSTRT,MEND
      IEND=M-1
      DO 530 II=BB,IEND
      R=IAM2(M,II)/IAM2(M,M)
      RCHK=DABS(R)
      RCHK=DSQRT(RCHK)
      DO 520 JJ=BB,DIM3
      ACHK=DABS(IAM2(JJ,M))
      ACHK=DSQRT(ACHK)
      PCHK=RCHK*ACHK
      IF(PCHK.LT.1.D-19)GO TO 520
      IAM2(JJ,II)=IAM2(JJ,II)-R*IAM2(JJ,M)
520   CONTINUE
530   CONTINUE
540   CONTINUE
      DO 570 M=MSTRT,MEND
      IEND=M-1
      DO 560 II=BB,IEND
      R=IAM2(II,M)/IAM2(M,M)
      RCHK=DABS(R)
      RCHK=DSQRT(RCHK)
      DO 550 JJ=BB,DIM3
      ACHK=DABS(IAM2(M,JJ))
      ACHK=DSQRT(ACHK)
      PCHK=RCHK*ACHK
      IF(PCHK.LT.1.D-19)GO TO 550
      IAM2(II,JJ)=IAM2(II,JJ)-R*IAM2(M,JJ)
550   CONTINUE
560   CONTINUE
570   CONTINUE
      IF((I+J).EQ.DIM)GO TO 573
      GO TO 575
```

```
573     CALL DET(IAM2,BB,I,POLY,NCOEFF,DETPQI)
        GO TO 1060
575     II=BB+I+J
        JJ=BB
        SROW=II
        LCOL=BB+I-1
580     IF(DABS(IAM2(II,JJ))-PINCH)610,610,590
590     IF(II.EQ.DIM3)GO TO 810
600     CALL SUB4(IAM2,IAM1,BB,DIM3,II,JJ,DIM3,LCOL)
        II=II+1
        JJ=JJ+1
        GO TO 580
610     IAM2(II,JJ)=0.
        IF(II.EQ.DIM3)GO TO 650
        WW=II+1
620     IF(DABS(IAM2(WW,JJ))-PINCH)640,640,630
630     CALL SUB1(IAM1,IAM2,II,WW,DETPQI,BB,DIM3)
        GO TO 600
640     IAM2(WW,JJ)=0.
        WW=WW+1
        IF(WW.GT.DIM3)GO TO 650
        GO TO 620
650     WW=JJ+1
660     IF(DABS(IAM2(II,WW))-PINCH)680,680,670
670     CALL SUB2(IAM1,IAM2,JJ,WW,DETPQI,BB,DIM3)
        GO TO 590
680     IAM2(II,WW)=0.
        WW=WW+1
        IF(WW.GT.LCOL)GO TO 690
        GO TO 660
690     IF(II.EQ.DIM3)GO TO 1080
        RR=II+1
        SS=JJ+1
700     IF(DABS(IAM2(RR,SS))-PINCH)720,720,710
710     CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
        CALL SUB2(IAM1,IAM2,JJ,SS,DETPQI,BB,DIM3)
```

```
      GO TO 600
720   IAM2(RR,SS)=0.
      IF(RR.EQ.DIM3)GO TO 760
      WW=RR+1
730   IF(DABS(IAM2(WW,SS))-PINCH)750,750,740
740   CALL SUB1(IAM1,IAM2,II,WW,DETPQI,BB,DIM3)
      CALL SUB2(IAM1,IAM2,JJ,SS,DETPQI,BB,DIM3)
      GO TO 600
750   IAM2(WW,SS)=0.
      WW=WW+1
      IF(WW.GT.DIM3)GO TO 760
      GO TO 730
760   WW=SS+1
770   IF(DABS(IAM2(RR,WW))-PINCH)790,790,780
780   CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
      CALL SUB2(IAM1,IAM2,JJ,WW,DETPQI,BB,DIM3)
      GO TO 600
790   IAM2(RR,WW)=0.
      WW=WW+1
      IF(WW.GT.LCOL)GO TO 800
      GO TO 770
800   IF(RR.EQ.DIM3)GO TO 1080
      RR=RR+1
      SS=SS+1
      GO TO 700
810   CALL SUB4(IAM2,IAM1,BB,DIM3,II,JJ,DIM3,LCOL)
      CALL SUB3(IAM2,IAM1,DETPQI,BB,DIM3,SROW,BB,DIM3,LCOL)
      JJ=BB+I+J
      II=BB
      LROW=BB+I-1
      SCOL=JJ
820   IF(DABS(IAM2(II,JJ))-PINCH)850,850,830
830   IF(JJ.EQ.DIM3)GO TO 1050
840   CALL SUB4(IAM2,IAM1,BB,DIM3,II,JJ,LROW,DIM3)
      II=II+1
      JJ=JJ+1
```

```
      GO TO 820
850   IAM2(II,JJ)=0.
      IF(JJ.EQ.DIM3)GO TO 890
      WW=JJ+1
860   IF(DABS(IAM2(II,WW))-PINCH)880,880,870
870   CALL SUB2(IAM1,IAM2,JJ,WW,DETPQI,BB,DIM3)
      GO TO 840
880   IAM2(II,WW)=0.
      WW=WW+1
      IF(WW.GT.DIM3)GO TO 890
      GO TO 860
890   WW=II+1
900   IF(DABS(IAM2(WW,JJ))-PINCH)920,920,910
910   CALL SUB1(IAM1,IAM2,II,WW,DETPQI,BB,DIM3)
      GO TO 830
920   IAM2(WW,JJ)=0.
      WW=WW+1
      IF(WW.GT.LROW)GO TO 930
      GO TO 900
930   IF(JJ.EQ.DIM3)GO TO 1080
      RR=II+1
      SS=JJ+1
940   IF(DABS(IAM2(RR,SS))-PINCH)960,960,950
950   CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
      CALL SUB2(IAM1,IAM2,JJ,SS,DETPQI,BB,DIM3)
      GO TO 840
960   IAM2(RR,SS)=0.
      IF(SS.EQ.DIM3)GO TO 1000
      WW=SS+1
970   IF(DABS(IAM2(RR,WW))-PINCH)990,990,980
980   CALL SUB1(IAM1,IAM2,II,RR,DETPQI,BB,DIM3)
      CALL SUB2(IAM1,IAM2,JJ,WW,DETPQI,BB,DIM3)
      GO TO 840
990   IAM2(RR,WW)=0.
      WW=WW+1
      IF(WW.GT.DIM3)GO TO 1000
```

```
      GO TO 970
1000  WW=RR+1
1010  IF(DABS(IAM2(WW,SS))-PINCH)1030,1030,1020
1020  CALL SUB1(IAM1,IAM2,II,WW,DETPQI,BB,DIM3)
      CALL SUB2(IAM1,IAM2,JJ,SS,DETPQI,BB,DIM3)
      GO TO 840
1030  IAM2(WW,SS)=0.
      WW=WW+1
      IF(WW.GT.LROW)GO TO 1040
      GO TO 1010
1040  IF(SS.EQ.DIM3)GO TO 1080
      RR=RR+1
      SS=SS+1
      GO TO 940
1050  CALL SUB4(IAM2,IAM1,BB,DIM3,II,JJ,LROW,DIM3)
      CALL SUB3(IAM2,IAM1,DETPQI,BB,DIM3,BB,SCOL,LROW,DIM3)
      DETPQI=DETPQI*((-1)**K)
      IF(K.EQ.I)GO TO 1100
      GO TO 7
1060  RETURN
1080  NCOEFF=0
      DETPQI=0.
      DO 1090 I=1,10
      POLY(I)=0.
1090  CONTINUE
      GO TO 1060
1100  POLY(1)=1.
```

```
      NCOEFF=1
      GO TO 1060
3000  FORMAT(' ','MATCON')
      END
      SUBROUTINE FFORM(DETPQI,POLY,NCOEFF,SSTRNG,M,
     1PTRX,VAL,MX,COEFFX,PROFSX)
      INTEGER*4 PTRX,PTR,SSTRNG(10),CNTR,PROFSX(40)
      REAL*8 DETPQI,POLY,VAL,COEFFX,PRDCT
      DIMENSION MX(40,18),COEFFX(40),VAL(40),POLY(10)
      GO TO 50
      TYPE 10,DETPQI,M,NCOEFF
10    FORMAT(' ','DETPQI= ',E10.3,'   M=',I2,'   NCOEFF
     1=',I2)
50    CONTINUE
      GO TO 60
      TYPE 20,(SSTRNG(I),I=1,8)
20    FORMAT(' ','SSTRNG=(',7(I2,','),I2,')')
      TYPE 30,(POLY(I),I=1,5)
30    FORMAT(' ','POLY=(',4(E10.3,','),E10.3,')')
      TYPE 35,DETPQI
35    FORMAT(' ','DETPQI= ',E10.3)
      GO TO 60
      TYPE 40,(VAL(I),I=1,5)
40    FORMAT(' ','VAL=(',4(E10.3,','),E10.3,')')
60    CONTINUE
      IF(NCOEFF.EQ.0)GO TO 500
      PTR=PTRX
      PRDCT=1.
      M1=M-1
      DO 400 CNTR=1,NCOEFF
      PROFSX(CNTR+PTR)=CNTR
      IF(M.EQ.1)GO TO 300
      DO 200 J=1,M1
      MX(CNTR+PTR,SSTRNG(J))=1
      IF(CNTR.EQ.1)PRDCT=PRDCT/VAL(SSTRNG(J))
200   CONTINUE
300   COEFFX(CNTR+PTR)=PRDCT*POLY(CNTR)*DETPQI
```

```
400     CONTINUE
        PTRX=PTR+NCOEFF
500     GO TO 950
        DO 600 I=1,15
        TYPE 700,(MX(I,J),J=1,8)
600     CONTINUE
700     FORMAT(' ',8(I1,1X))
        TYPE 800,(COEFFX(I),I=1,15)
800     FORMAT(' ','COEFFA=',5E10.3,/,5E10.3,/5E10.3)
        TYPE 900,(PROFSX(I),I=1,15)
900     FORMAT(' ','PROFSA=',15(I2,1X))
950     RETURN
1000    FORMAT(' ','FFORM')
        END
        SUBROUTINE SUB1(IAM1,IAM2,ROWI,ROWJ,DET,BB,DIM3)
        INTEGER*4 ROWI,ROWJ,BB,DIM3
        REAL*8 IAM1(20,20),IAM2(20,20),DET,SS1,SS2
        DO 100 II=BB,DIM3
        SS1=IAM1(ROWI,II)
        SS2=IAM2(ROWI,II)
        IAM1(ROWI,II)=IAM1(ROWJ,II)
        IAM2(ROWI,II)=IAM2(ROWJ,II)
        IAM1(ROWJ,II)=SS1
        IAM2(ROWJ,II)=SS2
100     CONTINUE
        DET=-DET
        RETURN
        END
        SUBROUTINE SUB2(IAM1,IAM2,COLI,COLJ,DET,BB,DIM3)
        INTEGER*4 COLI,COLJ,BB,DIM3
        REAL*8 IAM1(20,20),IAM2(20,20),DET,SS1,SS2
        DO 100 II=BB,DIM3
        SS1=IAM1(II,COLI)
        SS2=IAM2(II,COLI)
        IAM1(II,COLI)=IAM1(II,COLJ)
        IAM2(II,COLI)=IAM2(II,COLJ)
```

```
      IAM1(II,COLJ)=SS1
      IAM2(II,COLJ)=SS2
100   CONTINUE
      DET=-DET
      RETURN
      END
      SUBROUTINE SUB3(IAM1,IAM2,DET,BB,DIM3,SROW,
     1SCOL,LROW,LCOL)
      INTEGER*4 BB,DIM3,SROW,SCOL,DIM,RR,SS
      REAL*8 IAM1(20,20),IAM2(20,20),DET,R
      PINCH=1.0E-12
      RR=SROW-1
      SS=SCOL-1
      DIM=LROW-SROW+1
      DO 100 II=1,DIM
      IF(ABS(IAM1(II+RR,II+SS))-PINCH)60,60,70
60    IAM1(II+RR,II+SS)=0.
      GO TO 100
70    R=1./IAM1(II+RR,II+SS)
      DO 80 JJ=BB,DIM3
      IAM1(II+RR,JJ)=IAM1(II+RR,JJ)*R
      IAM2(II+RR,JJ)=IAM2(II+RR,JJ)*R
80    CONTINUE
      DET=DET/R
      IAM1(II+RR,II+SS)=1.
100   CONTINUE
      RETURN
      END
      SUBROUTINE SUB4(IAM1,IAM2,BB,DIM3,SROW,SCOL,LROW,LCOL)
      INTEGER*4 SROW,SCOL,BB,DIM3
      REAL*8 IAM1(20,20),IAM2(20,20),R
      RR=SROW+1
      SS=SCOL+1
      IF(SROW.EQ.LROW)GO TO 40
      DO 30 II=RR,LROW
      IF(ABS(IAM1(II,SCOL))-PINCH)25,25,10
```

```
10      R=IAM1(II,SCOL)/IAM1(SROW,SCOL)
        DO 20 JJ=BB,DIM3
        IAM1(II,JJ)=IAM1(II,JJ)-R*IAM1(SROW,JJ)
        IAM2(II,JJ)=IAM2(II,JJ)-R*IAM2(SROW,JJ)
20      CONTINUE
25      IAM1(II,SCOL)=0.
30      CONTINUE
40      IF(SCOL.EQ.LCOL)GO TO 80
        DO 70 II=SS,LCOL
        IF(ABS(IAM1(SROW,II))-PINCH)65,65,50
50      R=IAM1(SROW,II)/IAM1(SROW,SCOL)
        DO 60 JJ=BB,DIM3
        IAM1(JJ,II)=IAM1(JJ,II)-R*IAM1(JJ,SCOL)
        IAM2(JJ,II)=IAM2(JJ,II)-R*IAM2(JJ,SCOL)
60      CONTINUE
65      IAM1(SROW,II)=0.
70      CONTINUE
80      RETURN
        END
        SUBROUTINE DET(A,BB,N,POLY,NCOEFF,DETPQI)
        INTEGER*4 BB,OFFSET
        REAL*8 A,POLY,Q,P,B,TRACE,SUM
        DIMENSION A(20,20),P(10,10),Q(10,10),TRACE(10),
        IB(10),POLY(10)
        DO 1 I=1,10
        POLY(I)=0.
1       CONTINUE
        NCOEFF=0
        OFFSET=BB-1
        DO 3 I=1,20
        DO 3 J=1,20
        A(I,J)=-A(I,J)
3       CONTINUE
        TRACE(1)=0.
        DO 10 I=1,N
        TRACE(1)=TRACE(1)+A(I+OFFSET,I+OFFSET)
```

```
10    CONTINUE
      B(1)=-TRACE(1)
      IF(N.EQ.1)GO TO 76
      DO 20 I=1,N
      DO 20 J=1,N
      P(I,J)=A(I+OFFSET,J+OFFSET)
20    CONTINUE
      DO 75 L=2,N
      DO 40 I=1,N
      DO 40 J=1,N
      Q(I,J)=0.
      DO 40 K=1,N
      Q(I,J)=Q(I,J)+P(I,K)*A(K+OFFSET,J+OFFSET)
40    CONTINUE
      DO 50 I=1,N
      DO 50 J=1,N
      P(I,J)=Q(I,J)
50    CONTINUE
      TRACE(L)=0.
      DO 60 I=1,N
      TRACE(L)=TRACE(L)+P(I,I)
60    CONTINUE
      SUM=TRACE(L)
      LL=L-1
      DO 70 I=1,LL
      SUM=SUM+B(I)*TRACE(L-I)
70    CONTINUE
      FL=L
      B(L)=(-1./FL)*SUM
75    CONTINUE
76    DO 80 I=1,N
      POLY(I)=B(N+1-I)
80    CONTINUE
      POLY(N+1)=1.
      NCOEFF=N+1
      RETURN
```

```
      END
      SUBROUTINE NOCAP(IAM1,IAM2,BB,DIM3,POLY,NCOEFF,DET)
      INTEGER*4 BB,DIM3
      REAL*8 IAM1(20,20),IAM2(20,20),POLY,DET
      DIMENSION POLY(10)
      PINCH=1.E-12
      DO 100 I=1,10
      POLY(I)=0.
100   CONTINUE
      II=BB
110   IF(ABS(IAM2(II,II))-PINCH)150,150,120
120   IF(II.EQ.DIM3)GO TO 130
125   CALL SUB4(IAM2,IAM1,BB,DIM3,II,II,DIM3,II)
      II=II+1
      GO TO 110
130   DO 140 I-BB,DIM3
      DET=DET*IAM2(I,I)
140   CONTINUE
      POLY(1)=1.
      NCOEFF=1
      GO TO 200
150   IF(II.EQ.DIM3)GO TO 190
      JJ=II+1
160   IF(ABS(IAM2(JJ,II))-PINCH)180,180,170
170   CALL SUB1(IAM1,IAM2,II,JJ,DET,BB,DIM3)
      GO TO 125
180   IAM2(JJ,II)=0.
      JJ=JJ+1
      IF(JJ.GT.DIM3)GO TO 190
      GO TO 160
190   DET=0.
      NCOEFF=0
200   RETURN
      END
      SUBROUTINE SGNCHK(X,Y)
      REAL*8 X,Y
```

```
      Y=1.
      IF(X.LT.0)GO TO 10
      GO TO 20
10    X=-X
      Y=-Y
20    RETURN
      END
```

APPENDIX E

SOLVE.F4 PROGRAM LISTING

```
      EXTERNAL NEQN
      REAL*8 RAN,RPN(5),IPN(4),FLAT(20),XO(20),SGN(14),IPD(4),RPD(5),
     1NOISE,XL,XU,NEQN
      INTEGER*4 ULOL
      REAL*4 IFILE
      NOISE=0.D0
      TYPE 10
10    FORMAT('1','INPUT FILENAME=',$)
      ACCEPT 20,IFILE
      OPEN(UNIT=1,FILE=IFILE)
50    READ(1,200,END=600,ERR=700),(RPN(I),I=1,5)
      READ(1,200),(IPN(I),I=1,4)
      READ(1,200),(RPD(I),I=1,5)
      READ(1,200),(IPD(I),I=1,4)
      READ(1,300),(FLAT(I),I=1,20)
      READ(1,300),(XO(I),I=1,20)
      READ(1,400),(SGN(I),I=1,14)
      READ(1,500),XL,XU,LSUBY,ULOL
      CALL INT(RAN,XL,XU,IER,FLAT,XO,RPN,RPD,IPN,IPD,NEQN,
     1LSUBY,SGN,ULOL)
      NOISE=NOISE+RAN
      GO TO 50
600   NOISE=DSQRT(NOISE)
      TYPE 1000,NOISE
1000  FORMAT(' ','THE OUTPUT NOISE =     ',D10.3,'   VOLTS')
      CLOSE(UNIT=1,FILE=IFILE)
      CALL EXIT
700   TYPE 800
800   FORMAT(' ','*********READ ERROR*********')
      CALL EXIT
20    FORMAT(A5)
200   FORMAT(5D10.3)
300   FORMAT(20D10.3)
400   FORMAT(14D8.1)
500   FORMAT(2D10.3,2I2)
      END
```

```
      SUBROUTINE INT(Y,XL,XU,IER,FLAT,XO,RPN,RPD,IPN,
     1IPD,FCT,LSUBY,T,ULOL)
      REAL*8 Y,XL,XU,FLAT(20),XO(20),RPN(5),RPD(5),IPN(4),
     1IPD(4),FCT,T(14),AUX(13),H,HH,E,DELT2,P,HD,X,
     2SM,Q
      INTEGER*4 R,ULOL
      R=ULOL+LSUBY
      NDIM=13
      AUX(1)=.5DO*(FCT(XL,RPN,IPN,RPD,IPD,FLAT,XO,R,T)+
     1FCT(XU,RPN,IPN,RPD,IPD,FLAT,XO,R,T))
      H=XU-XL
      IF(H)10,10,2
2     HH=H
      E=1.656D-20/DABS(H)
      DELT2=0.DO
      P=1.DO
      JJ=1
      DO 7 I=2,NDIM
      Y=AUX(1)
      DELT1=DELT2
      HD=HH
      HH=.5DO*HH
      P=.5DO*P
      X=XL+HH
      SM=0.DO
      DO 3 J=1,JJ
      SM=SM+FCT(X,RPN,IPN,RPD,IPD,FLAT,XO,R,T)
3     X=X+HD
      AUX(I)=.5DO*AUX(I-1)+P*SM
      Q=1.DO
      J1=I-1
      DO 4 J=1,J1
      II=I-J
      Q=Q+Q
      Q=Q+Q
4     AUX(II)=AUX(II+1)+(AUX(II+1)-AUX(II))/(Q-1.DO)
```

```
      DELT2=DABS(Y-AUX(1))
      IF(I-5)7,5,5
5     IF(DELT2-E)10,10,7
7     JJ=JJ+JJ
10    Y=H*AUX(1)
      RETURN
      END
      REAL*8 FUNCTION NEQN(W,RPN,IPN,RPD,IPD,FLAT,XO,Y,T)
      REAL*8 W,RPN(5),IPN(4),FLAT(20),XO(20),IMAJN,IMAJD,
     1REALN,REALD,NMAGSQ,DMAGSQ,T(14),IPD(4),RPD(5)
      INTEGER*4 Y
      REALN=RPN(1)+T(1)*(RPN(2)*W)**2+T(2)*(RPN(3)*W)**4+
     1T(3)*(RPN(4)*W)**6+T(4)*(RPN(5)*W)**8
      IMAJN=IPN(1)*W+T(5)*(IPN(2)*W)**3+T(6)*(IPN(3)*W)**5+
     1T(7)*(IPN(4)*W)**7
      REALD=RPD(1)+T(8)*(RPD(2)*W)**2+T(9)*(RPD(3)*W)**4+
     1T(10)*(RPD(4)*W)**6+T(11)*(RPD(5)*W)**8
      IMAJD=IPD(1)*W+T(12)*(IPD(2)*W)**3+T(13)*(IPD(3)*W)**5+
     1T(14)*(IPD(4)*W)**7
      IF(DABS(REALN).LT.1.D-9)GO TO 100
      IF(DABS(IMAJN).LT.1.D-9)GO TO 100
      IF(DABS(REALD).LT.1.D-9)GO TO 100
      IF(DABS(IMAJD).LT.1.D-9)GO TO 100
      REALN=REALN*1.D-10
      IMAJN=IMAJN*1.D-10
      REALD=REALD*1.D-10
      IMAJD=IMAJD*1.D-10
100   CONTINUE
      NMAGSQ=REALN*REALN+IMAJN*IMAJN
      DMAGSQ=REALD*REALD+IMAJD*IMAJD
      NEQN=(NMAGSQ/DMAGSQ)*FLAT(Y)*(1+XO(Y)/W)
      RETURN
      END
```