NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VIRGINIA

TRONICS DIVISION INTERNAL REPORT NO. 217

A COMPUTER-AIDED ANALYSIS ROUTINE INCLUDING OPTIMIZATION FOR MICROWAVE CIRCUITS AND THEIR NOISE

DAN L. FENSTERMACHER

JULY 1981

NUMBER OF COPIES: 150

A COMPUTER-AIDED ANALYSIS ROUTINE INCLUDING OPTIMIZATION FOR MICROWAVE CIRCUITS AND THEIR NOISE

ABSTRACT

The program FARANT was developed as an aid in microwave circuit design for the Electronics Research Division at NRAO. It is programmed in BASIC and has been implemented on the VAX-11 as well as the HP 9845A desktop computer. Using many ideas from COMPACT (a commercially available program accessed through TYMSHARE, INC.), FARANT combines the strength of a large variety of analyses with the flexibility of subroutines that are completely controlled by the user. FARANT offers full frequency analysis of user-specified two-port networks, extensive noise analysis, outputs in various parameter sets for the composite two-ports and their noise, a powerful circuit optimization routine, and direct plotting capabilities on the HP 9845A. It is also flexible enough to incorporate the user's own BASIC programs for specially tailored analysis problems, graphics, and many other individual needs.

ACKNOWLEDGMENTS

For their guidance and continuing encouragement I must thank Dr. Sander Weinr and Dr. John Granlund of the Electronics Division in Charlottesville. The direction they gave to my efforts was gratefully accepted both for the overall support it provided as well as the numerous occasions where technical consultation became mandatory. Without their help the project could not have come into being. I would also like to thank Les Besser of Compact Engineering, Inc. for ideas and instruction in microwave circuit design methods.

A COMPUTER-AIDED ANALYSIS ROUTINE INCLUDING OPTIMIZATION FOR MICROWAVE CIRCUITS AND THEIR NOISE

Table of Contents

Table of Contents

I. USER'S MANUAL FOR FARANT

I.1 Introduction

The program FARANT is a steady-state a-c microwave circuit analysis routine that was developed to provide flexible approaches to designers' analysis problems. It can be described as a "cooperative" set of subroutines that the user controls and exploits, as opposed to an interactive program that asks the user for information during a run. In this sense FARANT is passive and vulnerable to the user's manipulations.

The organization of FARANT lets the user control his analysis by calling subroutines from within a designated area of the program. That area is itself a subroutine called "SUB Cktanalysis" and is placed at the end of FARANT for convenience. Almost all of the user's commands consist of statements he places within that subroutine, with exceptions noted in the individual command descriptions. Since any BASIC statements are allowed there, individual analysis approaches can be tailored to the user's requirements, and he can even add other subroutines to FARANT if he so desires.

This manual is intended to be a guide to using FARANT on the HP 9845A with special sections being devoted to usage on the VAX. The two versions of FARANT are logically identical but there are several differences in the specific details of their usage. For instance, BASIC syntax varies slightly between the two machines, file manipulations are quite different on the VAX, and the VAX is accessed through a terminal (an Apple computer being one possibility) whereas the HP 9845A is a stand-alone machine. This last difference is the reason that graphics, for example, are not yet available from the VAX version of the program, as they are on the HP 9845A.

I.2 Conventions and Definitions

Since FARANT is controlled by passing information to and from its subroutines, certain conventions are established to standardize the notation and information-sharing mechanisms. The units used throughout the program for subroutine arguments and printouts are:

Ohms	Giga-Hertz (10^9)
Milli-Mhos (10^{-3})	Inches
Nano-Henries (10^{-9})	Degrees (of arc)
Pico-Farads (10^{-12})	Degrees Kelvin (temperature)
Pico-Seconds (10^{-12})	

Two-port parameters, which are used to describe the behavior of two-terminal-pair circuits at a given frequency, are designated in five different sets or representations. As shown in the following, a number is used to identif each equivalent set of parameters [1]:

[A] = 1 (ABCD matrix)

[Z] = 2 (Impedance matrix)

[Y] = 3 (Admittance matrix)

[S] = 4 (Scattering matrix)

[T] = 5 (Transmission matrix)

The variable "Pset" is used in several subroutines to specify a certain type of network parameter set and takes on values from 1 to 5.

Similarly, the 4 real numbers which are needed to describe the behavior of the random noise generators within a two-port can be designated in many ways, all of which are equivalent. FARANT's subroutines use the variable "Nset" to

specify a specific noise parameter set according to the following:

(1)	=	R_n	G_n	$\|\rho\|$	ϕ
(2)	=	R_1	R_2	$\|\rho_v\|$	ϕ_v
(3)	=	G_1	G_2	$\|\rho_c\|$	ϕ_c
(4)	=	T_{min}	R_{opt}	X_{opt}	G_n
(5)	=	T_a	T_b	$\|\rho_{ab}\|$	ϕ_{ab}
(6)	=	T_{min}	T_b	$\|\Gamma_{opt}\|$	ϕ_{opt}
(7)	=	r_n	G_n	R_{cor}	X_{cor}
(8)	=	R_n	g_n	G_{cor}	B_{cor}

The definitions of these parameter sets can be found in section IV.4 and IV.5 of this document.

The statement which runs FARANT is the "CALL" statement which transfers control to a specific subroutine for specific operations to be performed. It has the form:

CALL element or function (list of parameters)

The elements are two-port circuits such as R-L-C circuits, transmission lines, controlled sources, etc. Functions do such things as to connect the elements, calculate their properties, print parameters for an overall circuit, and so fo The parameter list can consist of values, strings, variables, or numeric expressions of either sign. When variables are used, they need not have the same names that the subroutines use, but they must correspond in number and type to each parameter in the list. The following definitions apply to the parameters in the CALL statement:

two-port description: an array of dimension (6 X 4) that when passed to and from subroutines is denoted by a "two-port identifier," a

letter A - H followed by (*), e.g., D(*). When fully loaded it contains two-port parameters in rows 1 - 4 (see section IV.3), their label--"Pset"--in element (5,1), noise parameters in row 6, and their label--"Nset"--in element (5,2). (Elements (5,3) and (5,4) are never used.)

input parameters: those variables in a subroutine's parameter list which carry values to a subroutine. They can be previously-assigned variables (whose values will not be changed by the subroutine), constants, or numeric expressions.*

output parameters: those variables in a subroutine's parameter list which receive values from a subroutine. They must be actual variables if they are to receive their new values. (Constants or expressions cannot be assigned new values; their use causes the new values to be lost.) Thus, if a parameter is used for input and output it must be a variable, not a constant or expression.

Note that when a two-port identifier is used as input to a subroutine, its set of two-port or noise parameters may be changed, but they will still describe the same circuit.

I.3 User Commands

I.3.1 Frequency Specification

The user must specify the frequencies for analysis before calling the subroutines of FARANT. This can be done in a variety of ways using BASIC statements, but all must assign values (in GHz) to the variable F which is reserved for frequency. For instance,

```
F = 5
```

would specify 5 GHz as the only analysis frequency,

```
FOR F = 1 to 2  STEP .1
.
.
.
NEXT F
```

would set up a loop for analysis over a range of frequencies. Within a frequency loop such as this, the overall circuit can be built up, analyzed in various ways, and stored for later use at each frequency.

I.3.2 Two-Port Elements

R-L-C 2-Ports:

CALL Rlc(X(*),Type$,R,L,C,Place$,Tamb)

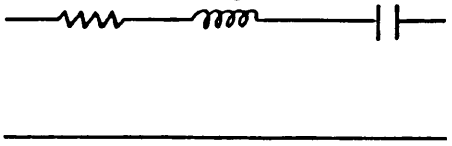
X(*) is a two-port identifier used for output and names the two-port which is created; a letter A through H followed by (*). All other parameters are used for input only.

Type$ and Place$ refer to the type of R-L-C (series or parallel) placed in series or parallel in the two-port. Thus, both parameters require either "S" or "P", and the quotes are necessary.

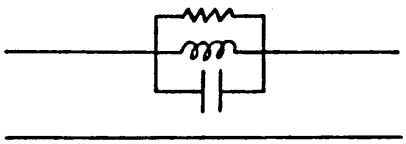
R,L,C are the values of resistance, inductance and capacitance in units of ohms, nano-Henries and pico-Farads. Positive or negative values are allowed, but a value of zero denotes the lack of that element in the two-port (which is sometimes equivalent to its having zero value and sometimes infinite.)

Tamb is the physical temperature in degrees Kelvin which determines the thermal noise properties of the two-port. It should be positive if R > 0 and noise analysis is desired. It should be 0 for all other ca

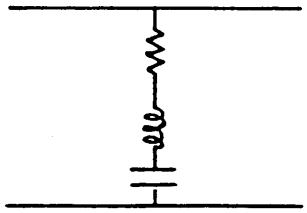
A "S"eries R-L-C placed in "S"eries



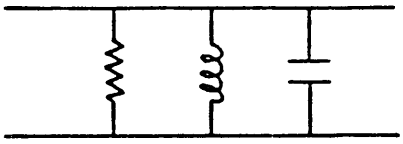
"P" in "S"



"S" in "P"



"P" in "P"



Lossless Transmission Lines (TEM):

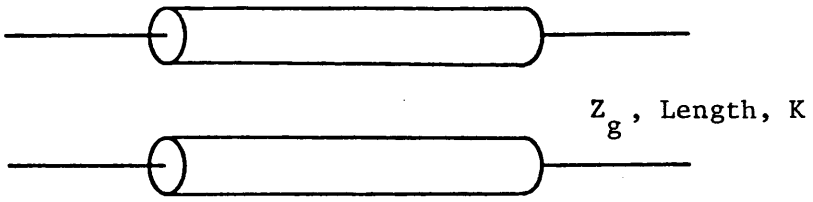
CALL Trline(X(*),Zg,Length,K)

X(*) is the two-port identifier for output; other parameters for input.

Zg is the (real) characteristic impedance of the lossless line $\sqrt{\frac{L}{C}}$.

Length is the physical dimension in inches.

K is the relative dielectric constant, $\frac{\varepsilon\mu}{\varepsilon_o\mu_o}$ for permeable dielectrics



Lossy Transmission Lines (TEM):

CALL Lossyline(X(*),Zg,Length,K,Cattn,Dattn,Fo,Tamb)

The first four parameters are as in the lossless case above.

Cattn is the attenuation in dB/inch that the line would have with only conductor losses.

Dattn is the same, but for the dielectric losses only.

Fo is the frequency at which Cattn and Dattn were determined, usually different from the current analysis frequency.

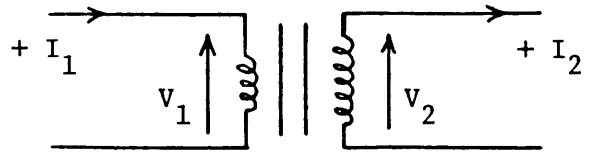
Tamb is the physical temperature used to determine the thermal noise. should be positive if noise analysis is desired, otherwise zero.

Ideal Transformers:

CALL Tf(X(*),Turns1,Turns2)

X(*) is the two-port identifier for output.

$\frac{\text{Turns1}}{\text{Turns2}}$ is the actual turns ratio and can be negative to reverse the polarity of port 2's current and voltage. The two numbers are used as input. Shown is the positive sense of I and V for a positive turns ratio:



Controlled Sources:

CALL Source(X(*),Control$,Source$,Gain,R1,R2,Delay)

X(*) is the two-port identifier for output. All other parameters are for input.

Control$ is either "C" or "V" for a current- or voltage-controlled source.

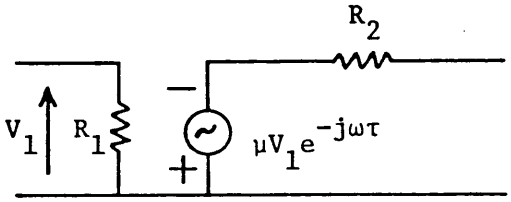
Source$ is also "C" or "V" for the source itself.

Gain is either μ, α, r_m, g_m where the trans-conductance g_m is given in mMhos. The convention is to have positive current into port 2 when shorted, but the Gain can, of course, be negative.

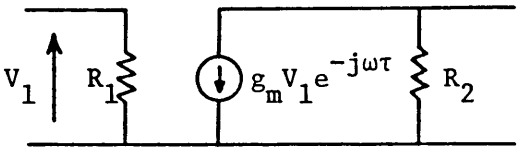
R1,R2 are the resistances at ports 1 and 2 and can be in the range 0 to 10^{10} or so, and all values, including zeroes, are interpreted numeric

Delay is the time lag between the control and the source, in pSec.

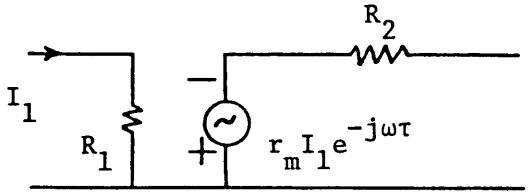
"V" controlled "V" source



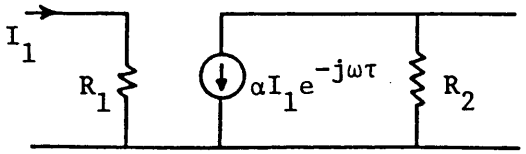
"V" controlled "C"



"C" controlled "V"



"C" controlled "C"



Measured Two-Port or Noise Parameters:

CALL Pread(X(*))

X(*) is a two-port identifier used strictly for output. Upon exit from SUB Pread, X(*) is loaded with the type of two-port parameters given by the data and has no noise representation.

In addition to the CALL, the user must store his measured data in the last lines of SUB Pread(X(*)) in DATA statements. The necessary values are:

Pset (1 to 5), "MPH" or "RI", Number of Frequencies

F_i and 8 values at that frequency, for each frequency.

Pset must correspond to the type of data that the user inputs and is not changed by the subroutine. "MPH" tells Pread to interpret each set of 8 values as magnitudes and phases (in degrees) for each parameter 11, 12, 21, 22; otherwise the 8 numbers are interpreted as real and imaginary parts of the 4 parameters. There is no limit to the number of frequencies, but only those frequencies which are stored can be used in subsequent analyses calling Pread, or else FARANT will terminate execution. The requirements for data are listed in a comment statement in SUB Pread just before the lines reserved for the data, so they can be viewed from the keyboard. The frequencies must be distinct, but need not be in order.

CALL Nread(X(*))

X(*) is a two-port identifier which is used for output. Although the two-port parameters in X(*) are not used by Nread, they should be defined before Nread is called since certain noise parameter transformations that FARANT performs require the two-port parameters.

As in Pread the user must store his noise data in the last lines of this subroutine in DATA statements. The requirements are shown in a comment statement there and consist of:

Nset (1 to 8), Number of Frequencies

F_i and 4 values at that frequency, for each frequency.

The 4 noise parameters for each of the 8 sets are given among the conventions in section I.2.

The frequencies need not be in order but must be unique. If the user calls Nread at a frequency for which there are no parameters, they are taken as zero and a warning is printed. Otherwise X(*) is loaded with the noise parameters in the given parameter set for the current analysis frequency.

CALL Nload(X(*),Nset,N1 or Tamb,N2,N3,N4)

X(*) is the two-port identifier used for output and should contain two-port parameters when Nload is called.

Nset is input as 0 to 8 where a zero value causes Nload to calculate and assign thermal noise to X(*) (which in that case must be a passive two-port), and values 1 - 8 to designate the type of noise parameters that follow.

N1 to N4 are input as a set of 4 noise parameters at the current analysis frequency to be assigned to X(*). If Nset = 0, then N1 is interpreted as a physical temperature and N2 to N4 have no effect.

This CALL functions similarly to a CALL of Nread except that here the data is passed directly to the subroutine instead of being stored in DATA statements. Since N1 to N4 are only used for input, they can be expressions and, for example, could serve to describe the non-thermal noise of an active two-port as a function of frequency.

I.3.3 Exchanging Ports and Creating Branch Elements

CALL Flip (X(*))

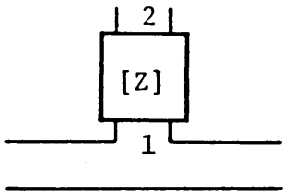
X(*) is a two-port identifier used for input and output, and after the CALL describes the same circuit as on input, including the noise, but with ports 1 and 2 reversed.

CALL Branch(X(*),Type$)

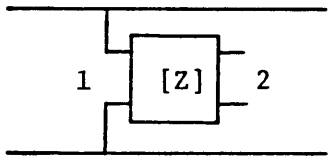
X(*) is a two-port identifier used for input and output. On output it describes the branch-element two-port created from port 1 of the input circuit and its noise, with port 2 left open.

Type$ is input as "S" or "P" to indicate a series or parallel branch as shown in the following:

"S"eries Branch



"P"arallel Branch



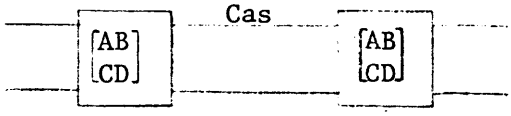
I.3.4 Cascading, Paralleling, and Putting 2-Ports in Series

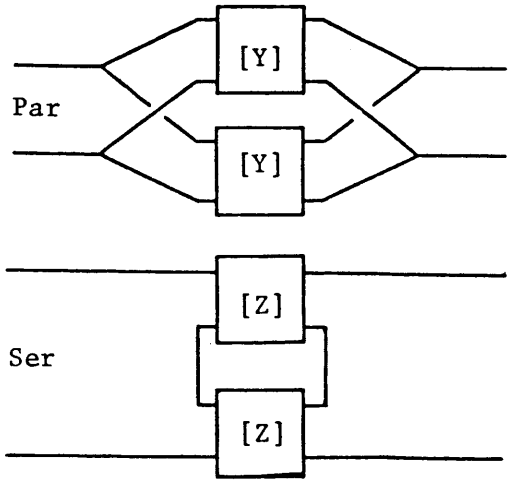
CALL Cas(X(*),A(*))

CALL Par(X(*),A(*))

CALL Ser(X(*),A(*))

X(*) and A(*) are two-port identifiers used for input, X(*) being used for output as well. The two circuits are connected as shown below and X(*) gets both the two-port and noise parameters of the composite two-port. (A(*) can be the same identifier as X(*), in which case a duplicate of X(*) is connected to itself.)





I.3.5 Transforming 2-Port and Noise Parameter Sets

CALL Mtrans(X(*),Pset)

CALL Ntrans(X(*),Nset)

X(*) is a two-port identifier used for input and output. SUB Mtrans will change its two-port parameters into the Pset type, leaving any noise representation alone. Calling Ntrans changes the noise parameters into the Nset type and may also change the two-port parameter set (if the noise parameters change to or from the 2 or 3 set).

Pset is input as 1 to 5 according to the 2-port parameter conventions described in section I.2.

Nset is input as 1 to 8 according to the noise conventions.

For the cases in which the two-port parameters are undefined and thus cannot be obtained, Mtrans quietly sets the variable Nogo = 1 and leaves X(*) with some other parameter set, not the one requested by Pset.

I.3.6 Saving Circuit Parameters

CALL Saveckt(X(*),Pset,Nset,Kfact)

X(*) is a two-port identifier used for input describing the circuit to be saved in the "data-base." (See section II.3.)

Pset is input as 1 to 5 to name the desired type of two-port parameters to be stored.

Nset is similarly input as 0 to 8, where a zero ignores any noise parameters in X(*) and stores zeroes instead. Using zero therefore saves analysis time.

Kfact is used for input and optionally for output as well. If its value is negative, no K-factor is calculated and a zero is stored; otherwise

the K-factor is calculated and stored in the data base. If Kfact is a non-negative valued variable rather than a constant, it receives the K-factor in addition to its being stored. This allows the user the option of using K directly from Saveckt, instead of having to get it from the data-base, and gives him full precision (12 digits) instead of the data-base's 6 digits.

Note that it is most accurate and efficient if Pset and Nset name the types of parameters asked for in later printouts because the data-base is kept only to 6 digits, but this is not required. However, in order to maintain consistent parameters in the data-base, Pset and Nset must be the same each time Saveckt is called, for any given run of FARANT. Saveckt also increments the variable Count -- the number of frequencies currently stored in the data-base -- to keep track of where the next storage position should be.

I.3.7 Noise Temperature and Gain Analysis

CALL Nperformance(X(*),Gtype,Rs,Xs,Rl,Xl,Gain,Tn)

X(*) is a two-port identifier used for input and names the circuit to be analyzed for gain and noise temperature. It should be the circuit which was used in calling Saveckt.

Gtype is input as 0 to 4 to specify the type of gain to be calculated. Zero omits any gain calculation, whereas 1 to 4 request transducer, power, available, and maximum available gain calculations respectively.

Rs,Xs are input as the real and imaginary parts of the source impedance driving the two-port.

Rl,Xl are input as the real and imaginary parts of the load impedance driven by the two-port.

Gain is used for output and receives the gain in dB provided Gtype = 1 to 4.

Tn is used for both input and output. It receives the noise temperature of the two-port driven from $R_s + jX_s$ provided $R_s > 0$, the two-port had noise parameters, and Tn was non-negative on input. A negative T_n on input thus suppresses the noise temperature calculation and saves time.

Nperformance should only be called subsequent to a CALL of Saveckt, and should use the same two-port X(*). It stores the source impedance in the data-base, as well as the gain and noise temperature if these were requested. The gain-type must be the same on each call of Nperformance, and will be the type produced in later printouts.

I.3.8 Reflections and Impedance Calculations

CALL Gammaz(Option,U,V,R,X)

Option is input as -2 to 2 to specify the following conversions:

-2	Z to Gamma (rectangular)
-1	Z to Gamma (polar)
0	Nothing
1	Gamma (polar) to Z
2	Gamma (rectangular) to Z

U,V are for input or output of the reflection coefficient gamma, i.e., U + jV (rectangular) or U exp(jπV/180) (polar), according to the value of Option.

R,X are for input or output of the impedance in rectangular form, i.e., Z = R + jX, according to the value of Option.

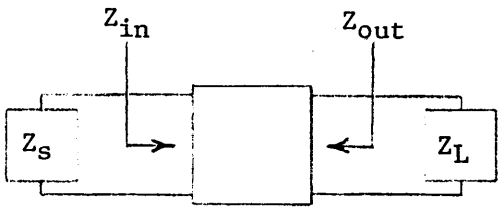
In making the Gamma ↔ Z conversion, the current Zo is used, e.g., 50 ohms. All real values of the 4 parameters are allowed including reflection coefficients whose magnitudes are greater than 1 and negative resistances.

CALL Zio(X(*),Rs,Xs,Rl,Xl,Rin,Xin,Rout,Xout)

X(*) is a two-port identifier used for input.

Rs,Xs,Rl,Xl are input as the real and imaginary parts of the source and load impedances attached to the two-port.

Rin,Xin,Rout,Xout are for output and receive the real and imaginary parts of the input and output impedances as shown:



I.3.9 Printing Circuit Parameters

CALL Prt(Pset,Nset)

Pset is input as +1 to +5 to produce a printout in Pset parameters of all the two-ports previously saved in the data-base by Saveckt. The data-base is also transformed if necessary into Pset parameters. Values of -1 to -5 suppress printing and simply transform the data-base. A zero suppresses both the printing and transforming of any two-port parameters.

Nset is similarly input as +1 to +8 to specify a noise parameter set for printing and transforming the data-base, or -1 to -8 to perform the transformation without the printing. A value of zero suppresses both.

This CALL allows the user to print either or both of 2 sets of network descriptions. For positive Pset values, the printout is:

Frequency X_{11} X_{12} X_{21} X_{22} K-factor

where "X_{ij}" stands for some two-port parameter, e.g., S_{21}, and is printed in polar form. Positive values of Nset produce a printout as follows:

Frequency Gain T_n 4 Noise Parameters R_s X_s

All of the two-port and noise parameters that were saved by calling Saveckt will be printed here; the frequencies need not be unique or ordered, nor must each set of parameters (i.e., line of printout) describe the same circuit. The K-facto gain, noise temperature, and noise parameters, of course, must have been properly requested earlier in order that non-zero values be printed here.

The feature of being able to re-store network data without printing it is useful when making plots from the data-base. (See sections II.3 and II.4.) Note that time is saved and precision preserved, however, by requesting the same Pset and Nset parameters that were saved by the call of Saveckt.

I.3.10 Plotting on a Smith Chart

CALL Smith(Xmin,Xmax,Ymin,Ymax)

Xmin...Ymax are input as the left and right-hand, bottom and top coordinates respectively of the extremes of the Smith chart desired.

This CALL produces a Smith chart of any size or any portion thereof. The largest full chart, for example, is obtained using parameters -1, 1, -1, 1, but any part of the chart can be expanded (to the full size of the plotter) by giving coordinates with magnitudes less than 1. Magnitudes greater than 1 produce a large plotting area, and thus a smaller Smith chart. Previous graphics are not

erased by this CALL; the plotting area is only scaled to the given specifications and overlaid with a Smith chart. Note that the scaling is always done so as to maintain a circular chart no matter how "oblong" the dimensioning parameters are.

CALL Splot(I,J)

I,J are input as 1 or 2 to indicate the subscripts of the S-parameter to be plotted.

This CALL assumes that one circuit was analyzed at many frequencies when calling Saveckt. Splot produces a plot of that circuit's specified S-parameter over the frequency range stored and puts tic marks at those frequencies. Note that in doing so it first transforms the data-base's two-port parameters into the [S] parameters by calling Prt(-4,0).

I.3.11 Optimization

CALL Optimize(N,X(*))

N is input as the number of variables on which to operate when minimizing the user's objective function. This will be explained below.

X(*) is used for input and output and is not a two-port identifier. It is a one-dimensional vector which on input contains initial guesses for the N variables, and on output contains the values of those variables which the optimization subroutine found to minimize the objective function.

This CALL exists in only one place, i.e., in the subroutine immediately preceding Cktanalysis called Farstart. (See section IV.1 for a flow diagram of FARANT's CALLing sequences, including the CALL of Optimize.) In turn, SUB Optimize calls only one other subroutine namely Cktanalysis:

SUB Cktanalysis(X(*),Fvalue,Opt)

X(*) is used for input to Cktanalysis and is the same vector of variables that is used by the Optimize subroutine. These variables determine the objective function.

Fvalue is an output of Cktanalysis set by the user's program statements to the value of the objective function given the values of the variables in X(*).

Opt is input to Cktanalysis as a flag having a value of either 0 or 1. When equal to 1, Cktanalysis is being called only to provide the calling subroutine -- SUB Optimize -- with a value of the objective function. A value of 0 means that Cktanalysis is being called with either the initial or final X(*) values and should therefore produce the output that the user desires.

The Objective Function:

FARANT's optimization capability is set up to minimize the value of a function of several variables, i.e., X(1), X(2)...X(N). The user uses these variables in any way he desires to define his function, and assigns the function's value to the variable Fvalue in SUB Cktanalysis. The variables are typically circuit-element values such as resistances, inductances, capacitances, transmission line lengths, transconductances, and so forth -- values of parameters passed to FARANT's subroutines. For example, CALL Rlc(B(*),"S",X(1),0,X(4),"P",0) uses X(1) and X(4) to define a R-L-C element. But there is no restriction on the variables except that they have a differentiable effect on the objective function. The objective function is some net measure of performance that the user wants to optimize numerically by making it as small as possible. It should be set up to have a minimum greater than or equal to zero, so a sum of squared terms is a typical choice. It can use any of the analysis results of FARANT, e.g., gain, noise temperature, $|S_{11}|^2$, K-factor, etc. and/or any user-derived results. It can be a combination of performance criteria at several frequencies in a given

band. In other words, the user can tailor the objective function to any form he likes, and SUB Optimize will find a minimum of that function. The shortcomings of this method are:

1) extensive computation time (proportional to the number of frequencies used in determining Fvalue, and to the number of variables).

2) the possibility of finding a local rather than a global minimum.

By exercising discretion in building up the objective function, and using care when picking an initial guess, the extent of these shortcomings can be minimized.

Initial Guesses:

In order for the optimization to begin, initial values of each variable must be assigned by the user to X(*). To do this, one must put the number of variables and their initial values in the data statement on line 6030 of SUB Farstart. This line is clearly marked with a comment statement to this effect. Zeroes must not be used, but the variables can be in any units, be positive or negative, and be of any relative size. The optimization performs best, however, when the initial guesses are not too different in magnitude from the values that will indeed produce a minimum, so some care is required to produce reasonable initial guesses.

Getting Under Way:

SUB Farstart is set up to structure the flow of optimization automatically. Its default is simply to call Cktanalysis (with the Opt flag set to zero) for normal (non-optimizing) analysis procedures. To run optimization, the user needs only to change the comment: ! CALL Optimize(N,X(*)) in line 6040 into an executable statement by deleting the exclamation point. The flow of control of FARANT then procedes as follows:

```
CALL Cktanalysis(X(*),0,0)        for initial-guess analysis
```

CALL Optimize(N,X(*))	to find the best values of X(*)
CALL Cktanalysis(X(*),0,0)	this time called from SUB Optimize (with its best values) for final analysis

Using the "Opt" Flag:

The last parameter passed to SUB Cktanalysis is the Opt flag, having a value 0 or 1. When FARANT is run using the optimizing capability, Cktanalysis is called for two different reasons, corresponding to the values of Opt. When Opt = 1, SUB Optimize is requesting a value of the objective function and the user should have Cktanalysis set up to perform the minimal number of calculations needed to define that function and put its value in the variable Fvalue. When Opt = 0, Cktanalysis is being called with either the initial values in X(*) or the final values. In this case, Cktanalysis can be set up to do more extensive procedures such as making plots, printing parameters, etc. One can make the distinction by using conditional statements with the Opt variable as a true/false antecedent, for example:

```
IF Opt THEN SUBEXIT
or
IF Opt = 0 THEN CALL Prt(...
```

By appropriate use of the Opt flag, one can get a fancy pre- and post-optimized output without slowing the intervening optimization process with unnecessary computation.

Output:

During the optimization procedure the values of the variables in X(*) are continuously changing, and the function value is decreasing. At each improved function value, SUB Optimize produces a listing of the updated X(*) values, the corresponding function value, and the sensitivity of the function

to separate changes in each variable. These sensitivities are in units (functi change) per (factor-of-e change in a given variable); specifically

$$\text{"sensitivity"} = |x_i| \frac{dF}{dx_i}$$

and each sensitivity should go to zero as the optimization process settles on a minimum. At completion, the optimization subroutine will print a summary of initial and final variables and their function values. It then calls Cktanalys passing it the last X(*), and uses an Opt flag of zero to allow Cktanalysis to produce its final output.

I.4 Summary: FARANT's Subroutines and Their Parameters in Program Order

```
SUB Mtrans(X(*),Pset)
SUB Ntrans(X(*),Nset)
SUB Rlc(X(*),Type$,R,L,C,Place$,Tamb)
SUB Source(X(*),Control$,Source$,Gain,R1,R2,Delay)
SUB Trline(X(*),Zg,Length,K)
SUB Tf(X(*),Turns1,Turns2)
SUB Lossyline(X(*),Zgo,Length,K,Cattn,Dattn,Fo,Tamb)
SUB Flip(X(*))
SUB Branch(X(*),Type$)
SUB Nload(X(*),Nset,N1,N2,N3,N4)
SUB Polar(X,Y)
SUB Cas(X(*),A(*))
SUB Par(X(*(,A(*))
SUB Ser(X(*),A(*))
SUB Saveckt(X(*),Pset,Nset,Kfact)
SUB Nperformance(X(*),Gtype,Rs,Xs,R1,X1,Gain,Tn)
```

```
SUB Prt(Pset,Nset)
SUB Smith(Xmin,Xmax,Ymin,Ymax)
SUB Splot(I,J)
SUB Gammaz(Option,U,V,R,X)
SUB Zio(X(*),Rs,Xs,Rl,Xl,Rin,Xin,Rout,Xout)
SUB Optimize(N,X(*))
SUB Nread(X(*))
SUB Pread(X(*))
SUB Farstart (no passed parameters)
SUB Cktanalysis(X(*),Fvalue,Opt)
```

I.5 Example Using Optimization

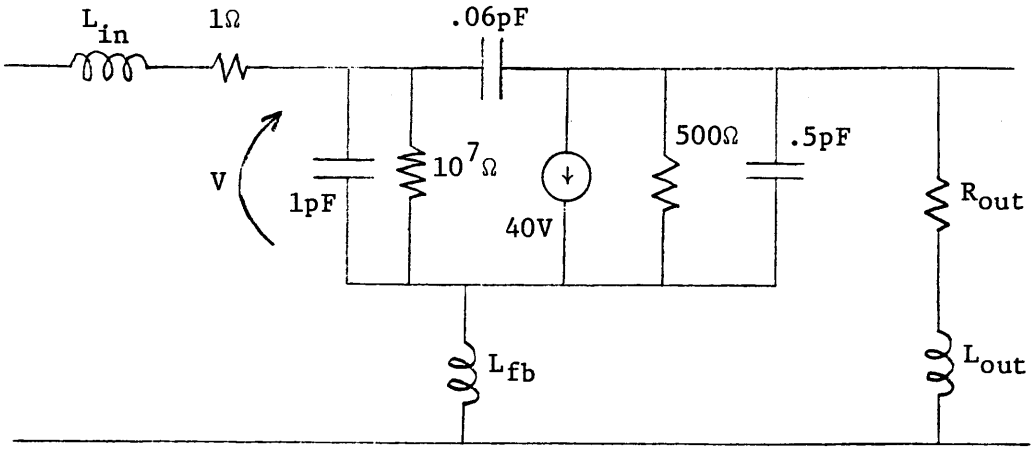
On the following pages is shown a sample listing and output from a demonstration run of FARANT that uses the optimization capability. The listing starting at line 6000 shows the user's portion of FARANT -- the initial guesses in line 6030, the CALL of Optimize in 6040, and the entire circuit description in lines 6145-6210. The purpose of this particular run was to optimize a combination of performance criteria at 1.6 GHz. The gain and input return loss were to be maximized, the noise temperature was to be minimized and the K-factor was to be greater than unity. For this purpose a function was constructed consisting of a sum of four positive terms, one for each of the performance criteria, whose values were close to 1 at a desirable level of performance:

$$\text{Fvalue} = \frac{25}{|S_{21}|^2} + 10|S_{11}|^2 + \frac{T_n}{50} + e^{10(1-K)}$$

Four circuit elements were allowed to vary in order to minimize this function, but their variation was constrained in the following way:

Variable	Constraint	Initial Value	Constraining Function	Initial
L_{in}	$\pm$ any value	15 nH	$L_{in} = X(1)$	15
L_{fb}	$.2 < L_{fb} < 2$	.466 nH	$L_{fb} = \frac{\arctan(X(2))}{100} + 1.1$	-2
R_{out}	$R_{out} > 10$	30 ohms	$R_{out} = 10 + e^{X(3)}$	3
L_{out}	$L_{out} > 0$	25 nH	$L_{out} = X(4)^2$	5

The final two-port circuit identified by A(*) in line 6215 looks like the following:



In the printout the following items can be observed:

1) The Opt flag was used to obtain a listing of parameters at several frequencies while calculating Fvalue only at 1.6 GHz.

2) Each of the 4 performance criteria actually did improve with optimizatio but did so by very different amounts.

3) The sensitivities went smoothly to zero as the optimization came to its completion. This signifies ample precision in the calculation of Fvalue.

4) Approximately 30 gradient steps were taken. To obtain 4 sensitivities and 1 function evaluation per step this required about 150 calls of Cktanalysis during optimization. Total time for this run on the HP 9845 was about 10 minutes

```
CALL "FSHOPT"

  "DEMO",6000

T 6000
0 SUB Farstart                    !USER'S CONTROL OF FARANT BEGINS HERE #######
5 OPTION BASE 1
0 DIM X(20)                       !IF N>20, MUST DIM [X] LARGER
5 READ N                          !# OF PARAMETERS TO OPTIMIZE
0 REDIM X(N)
5 READ X(*)                       !FOR INITIAL GUESSES ONLY WHEN OPTIMIZING
0 DATA 4, 15, -2, 3, 5            !PUT N, INITIAL GUESSES HERE (USE NO ZEROS
5 CALL Cktanalysis(X(*),0,0)      !FOR PRE-OPTIMIZED ANALYSIS; THE DEFAULT
0 CALL Optimize(N,X(*))           !USE THIS STATEMENT TO DO OPTIMIZATION
5 SUBEND
0 SUB Cktanalysis(X(*),Fvalue,Opt)                                  !##########
5 OPTION BASE 1
0 COM Nogo,Zo,F,Count,SHORT Dat(51,18)  !(DAT) HOLDS FREQ, CKT AND NOISE DATA
5 DIM A(6,4),B(6,4),C(6,4),D(6,4),E(6,4),F(6,4),G(6,4),H(6,4)
0 FIXED 3
5 Count=Nogo=0                    !#FREQS CURRENTLY STORED IN DATA BASE
0 Zo=50
5 DEG                             !DEFAULT FOR TRIG FUNCTIONS IS DEGREES
0 REM USER'S PROGRAM SHOULD BEGIN IN THE FOLLOWING LINES . . .
5 ! ---------------------------------------------------------------------
0 Lin=X(1)                        !DEFINING THE 4 CKT ELEMENTS IN TERMS OF
5 Lfb=ATN(X(2))/100+1.1           !X(1) . . . X(4)
0 Rout=10+EXP(X(3))
5 Lout=X(4)^2
0 F=1.6
5 !
0 IF Opt THEN 6145
5 PRINT "Lin =";Lin,"Lfb =";Lfb,"Rout =";Rout,"Lout =";Lout
0 FOR F=1.4 TO 1.8 STEP .1                !ANALYZE BAND OF FREQUENCIES
5 CALL Rlc(A(*),"S",1,Lin,0,"S",300)      !INPUT INDUCTOR (LOSSY)
0 CALL Rlc(B(*),"S",0,0,1,"P",0)          !GATE CAPACITANCE
5 CALL Source(C(*),"V","C",40.1E7,500,0)  !VOLTAGE CONTROLLED CURRENT SOURCE
0 CALL Rlc(D(*),"S",0,0,.5,"P",0)         !DRAIN CAPACITANCE
5 CALL Rlc(E(*),"S",0,0,.06,"S",0)        !INTERNAL FEEDBACK CAPACITANCE
0 CALL Rlc(F(*),"S",0,Lfb,0,"P",0)        !SOURCE-LEAD (FEEDBACK) INDUCTANCE
5 CALL Rlc(G(*),"S",Rout,Lout,0,"P",300)  !OUTPUT TUNING INDUCTANCE
0 CALL Par(C(*),E(*))
5 CALL Cas(B(*),C(*))
0 CALL Cas(B(*),D(*))
5 CALL Nload(B(*),4,50,70,200/F,3)        !Tmin, Ropt, Xopt, Gn FOR THE FET
0 CALL Ser(B(*),F(*))
5 CALL Cas(A(*),B(*))
0 CALL Cas(A(*),G(*))
5 CALL Saveckt(A(*),4,4,K)                !SAVE [S] AND THE #4 NOISE REP.
0 CALL Nperformance(A(*),1,50,0,50,0,Gt,Tn)  !TRANSDUCER-GAIN, AND NOISE TEMP
5 CALL Mtrans(A(*),4)                     !GET [S] FOR THE CKT'S |S11|,|S21|
0 !
5 G=25/(A(3,1)^2+A(3,2)^2)                !MINIMIZE: 25/|S21|^2
0 N=Tn/50                                 !MINIMIZE: Tn/50
5 M=10*(A(1,1)^2+A(1,2)^2)                !MINIMIZE: 10*|S11|^2
0 S=EXP(10*(1-K))                         !MINIMIZE: EXP[10*(1-K)]
5 Fvalue=G+N+M+S                  !ADD CONTRIBUTIONS TO THE ERROR FUNCTION
0 IF Opt THEN SUBEXIT             !FOR OPTIMIZATION, WE NEED ONLY DEFINE Fvalue
5 !
0 IF F<>1.6 THEN 6285
5    PRINT LIN(1);"MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6):";G;N;M;S
0    PRINT "Fvalue =";Fvalue
5 NEXT F
0 CALL Print(4,4)                         !PRINT [S] AND #4 NOISE PARAMETERS
5 SUBEND
```

RUN

Lin = 15.000 Lfb = .466 Rout = 30.086 Lout = 25.000

MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6): 2.257 1.602 6.414 14.135
Fvalue = 24.407

[S] PARAMETERS IN MAGNITUDE AND PHASE

	11		12		21		22		K
FREQ	MAG	ANG	MAG	ANG	MAG	ANG	MAG	ANG	FA
1.400	.6650	106.8	.0446	-4.6	4.7422	59.4	.6055	-17.7	.
1.500	.7403	86.3	.0375	-12.4	3.9709	48.0	.5929	-18.7	.
1.600	.8003	72.3	.0321	-17.1	3.3281	39.2	.5919	-20.0	.
1.700	.8492	62.3	.0272	-19.3	2.8135	32.1	.5965	-21.6	.
1.800	.8739	54.8	.0232	-19.4	2.4048	26.4	.6033	-23.6	.

TRANSDUCER-GAIN WAS REQUESTED, WHICH DEPENDS ON: Zsource, [S], Zload

NOISE PERFORMANCE PARAMETERS

FREQ	G (dB)	Tn	Tmin	Ropt	Xopt	Gn	Rs	Xs
1.400	13.52	65.24	56.10	71.78	7.00	3.01	50.00	0.0
1.500	11.98	66.92	56.07	71.80	-12.15	3.01	50.00	0.0
1.600	10.44	80.08	56.03	71.82	-30.09	3.00	50.00	0.0
1.700	8.99	102.79	55.99	71.84	-47.06	3.00	50.00	0.0
1.800	7.63	133.60	55.96	71.87	-63.20	2.99	50.00	0.0

INITIAL VALUES OF VARIABLES ARE:
15.0000 -2.0000 3.0000 5.0000

INITIAL FUNCTION VALUE = 24.4072

SENSITIVITIES: 16.6072 -61.2015 -32.1314 65.0214

THESE ARE INITIAL RELATIVE SENSITIVITIES OF THE VARIABLES (|V|*dFval/dV)

STEP# 1 [X]: 14.9115 .4459 3.8561 3.9606 Fval: 8.5968

SENSITIVITIES: 19.2553 -.1063 1.9734 -2.2913
STEP# 2 [X]: 13.6217 .8393 3.4015 4.4699 Fval: 6.7133

SENSITIVITIES: 14.7499 .0405 .4571 -.2384
STEP# 3 [X]: 8.6001 1.1501 2.8281 4.6855 Fval: 4.1817

SENSITIVITIES:-1.4312 .5492 -.6149 1.4436
STEP# 4 [X]: 9.3126 .9194 3.0780 4.4125 Fval: 3.9625

SENSITIVITIES: .5471 .3906 -.0187 .1336
STEP# 5 [X]: 9.0504 .7075 3.2504 4.1337 Fval: 3.9123

SENSITIVITIES:-.2195 .2986 .5402 -.9395
STEP# 6 [X]: 9.1026 .6393 3.2471 4.1561 Fval: 3.0987

NSITIVITIES: .0016 -.3000 .0014 -.0043
EP# 22 [X]: 9.0827 .1323 1.5505 3.8651 Fval: 3.6841

NSITIVITIES:-.0001 -.0000 .0005 -.0030
EP# 23 [X]: 9.0830 .1320 1.5321 3.8624 Fval: 3.6841

NSITIVITIES:-.0000 -.0000 -.0000 .0005
EP# 24 [X]: 9.0832 .1323 1.5268 3.8605 Fval: 3.6841

EP# 25 [X]: 9.0831 .1321 1.5294 3.8614 Fval: 3.6841

NSITIVITIES:-.0001 .0000 -.0000 .0001
EP# 26 [X]: 9.0831 .1321 1.5289 3.8613 Fval: 3.6841

NSITIVITIES:-.0000 .0000 -.0000 .0000
EP# 27 [X]: 9.0832 .1321 1.5286 3.8611 Fval: 3.6841

EP# 28 [X]: 9.0831 .1321 1.5286 3.8612 Fval: 3.6841

NSITIVITIES: .0000 .0000 .0000 .0000
EP# 29 [X]: 9.0832 .1321 1.5284 3.8611 Fval: 3.6841

TIMIZATION HAS TERMINATED AFTER 39 STEPS.

IT. Fval = 24.40722 INIT. VALUES:
5.00000 -2.00000 3.00000 5.00000

NAL Fval = 3.68412 FINAL VALUES:
.08315 .13209 1.52859 3.86119

n = 9.083 Lfb = 1.175 Rout = 14.612 Lout = 14.909

ASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6): 1.897 1.498 .236 .053
alue = 3.684

[S] PARAMETERS IN MAGNITUDE AND PHASE

	11		12		21		22		K
FREQ	MAG	ANG	MAG	ANG	MAG	ANG	MAG	ANG	FACT
1.400	.2539	-112.4	.0409	97.5	3.9816	105.1	.7156	14.9	1.25
1.500	.1603	-149.4	.0447	96.9	3.8315	95.1	.7196	10.0	1.20
1.600	.1535	153.5	.0489	96.1	3.6301	85.6	.7240	5.4	1.23
1.700	.2271	116.8	.0535	95.1	3.3970	76.8	.7286	.9	1.22
1.800	.3169	98.0	.0583	93.8	3.1507	68.7	.7331	-3.4	1.14

TRANSDUCER-GAIN WAS REQUESTED, WHICH DEPENDS ON: Zsource, [S], Zload

NOISE PERFORMANCE PARAMETERS

FREQ	G (dB)	Tn	Tmin	Ropt	Xopt	Gn	Rs	Xs
1.400	12.60	115.38	57.24	72.16	53.58	2.98	50.00	0.00
1.500	11.67	90.34	57.16	72.22	37.83	2.97	50.00	0.00
1.600	11.20	74.92	57.07	72.30	23.28	2.96	50.00	0.00
1.700	10.62	67.18	56.99	72.38	9.72	2.95	50.00	0.00
1.800	9.97	65.67	56.90	72.47	-3.00	2.94	50.00	0.00

II. DETAILED USER INSTRUCTIONS

II.1 The Reference Z_o for S-Parameters

The reference or normalization impedance for all S-parameters is assigned in Cktanalysis a value of 50 ohms when FARANT is run. The user can change this, however, by re-assigning the variable Z_o or by editing line 6090 of Cktanalysis. The use of transformers also comes in handy for changing the reference of a set of S-parameters measured in other than a 50-ohm system. For instance, after S-parameters are read into a matrix by the subroutine Pread, they can be changed to another reference impedance by sandwiching the 2-port between ideal transformers. The impedance ratio--the square of the turns ratio--must transform the relative impedance from Z_{ref} of the initial parameters (next to the device) to Z_o of the new parameters at the outside ports. Transformers can arbitrarily normalize the input and output to any line impedance that is desired.

Perhaps a simpler way of changing the Z_{ref}, however, is to use the subroutine Mtrans to change from [S] parameters to either [A], [Y] or [Z], with Z_o temporarily set equal to Z_{ref} of the initial device parameters. The variable Z_o is used by Mtrans as a common storage location for the normalization impedance. Thus, the statements CALL Pread(A(*)), Z_o = 73, CALL Mtrans(A(*),2), Z_o = 50 would change the 73 ohm S-parameters in A(*) to [Z] parameters and maintain a consistent 50 ohm system.

II.2 Creating Storage Space for More Elements

In line 6075, Cktanalysis sets up storage matrices of dimension (6 X 4) for 8 elements named A through H. There is nothing special about the number 8 or those particular letters, and the user should feel free to add to the list or

change the names according to his desires. Each dimensioned matrix takes about 200 bytes of memory, whether or not it is used in later statements. Two-port identifiers used in Cktanalysis must, of course, use the same names as these matrices.

The data-base matrix Dat(51,18) is not so flexible, however. The structure and size of the data-base is taken for granted by many of FARANT's subroutines and should not be changed unless a thorough editing of FARANT accompanies that change. Thus, the user can expect trouble if he tampers with the data-base.

II.3 Using the Data-Base

Although its structure should not be changed, the data-base can be used freely for plotting, making tables, and obtaining circuit parameters. The first 50 rows of Dat(51,18) all contain the items shown below. Three labels are kept in row 51 to describe the parameters in the first 50 rows. These labels are: Pset in element (51,1), Nset in element (51,2), and Gtype in element (51,3). As discussed in section I.3.9, the Prt subroutine can transform the data-base's two-port or noise parameter sets, but at any one time the entire data-base must be described by these labels. The 18 items stored in each row are the following:

1	2,3	4,5	6,7	8,9	10 - 13	14	15	16	17,18
F	X_{11}	X_{12}	X_{21}	X_{22}	4 Noise #'s	T_n	G	K	R_s,X_s

Each time Saveckt is called to store circuit parameters in the data-base, the variable Count is incremented by 1. This variable is held in COMmon storage (as is the data-base and some other variables -- see section IV.2) and designates the number of rows of the data-base that have been assigned by calling Saveckt. Thus, when making his own plots or tables, Count tells the user the number of items he has to work with.

One point that is worth emphasizing is that the data-base on the 9845 is stored using short precision which keeps about 6 digits of accuracy. This is sufficient for graphs and listings of circuit data but not precise enough for optimization purposes. When optimizing, one should take data directly from the two-port matrices or from the subroutine pass-parameters, e.g., for gain, noise temperature, and K.

It must also be noted that while the units for input to the subroutines and printouts are given in section I.2, the data-base and the two-port matrices use a slightly different set of units which can be summarized as follows:

Internal Calculations and Storage	Input/Printout
Mhos	Milli-Mhos
Correlations (of Noise Sources)	Correlation Coefficients
Rectangular Coordinates	Polar Coordinates

All other units are those given in section I.2.

II.4 Alternate Plotting and Analysis

Since the user can put any BASIC statements he likes into SUB Cktanalysis, there is ample freedom to generate special graphs or perform alternate kinds of analyses. On the 9845 the following will plot the reactive part of Z_{22} from a circuit that was stored using Saveckt over a range of frequencies between 1 and 2 GHz:

```
CALL Prt(-2,0)      - to transform the data-base to [Z]
GRAPHICS            - to display the plotting on the CRT
GCLEAR              - to erase previous graphs
```

```
LORG 5                          - to center the labels
LINE TYPE 1                     - to plot in solid lines
SCALE 1,2,-300,300              - 1 to 2 GHz, ± j300 ohms
AXES .1,50,1,0,5,2              - axes with tic marks
MOVE Dat(1,1),Dat(1,9)          - set the pen at the 1st point
FOR I = 1 TO Count
     PLOT Dat(I,1),Dat(I,9)     - plots a line to F,X22
     LABEL USING "K";"X22"      - "X22" is labelled and
                                  centered correctly
     NEXT I
LETTER                          - allows user to label his graph
```

It is possible that the user would like analysis over a specialized frequencies not specified by a simple FOR-NEXT loop. Frequency bands, example, can be specified by a loop construction as follows:

```
FOR I = 1 to 2                  - two bands
READ F1, F2, Delta              - initial F, final F, and increment
DATA 0, 10, 1, 15, 100, 5
FOR F = F1 to F2 STEP Delta     - 0 to 10 GHz by 1
                                - 15 to 100 GHz by 5


NEXT F
NEXT I
```

A logarithmic variation of frequency is achieved, for example, by:

```
M = (F2/F1)^1/10
F = F1
FOR I = 0 to 10
    .
    .
    .
    F = M * F
    NEXT I
```

Using the READ-DATA statements in combination with FOR-NEXT loops, one can tailor a frequency specification in any way he may desire.

Many more complicated tasks can also be programmed by the user in Cktanalysis or in separate subroutines that the user wants to add to FARANT. These tasks may include, for example, algebraic or matrix calculations, interactive input of decisions or data, special printouts, microstrip or waveguide analysis, data storage on magnetic tape, etc. The combination of user-defined procedures and FARANT's subroutines constitutes a versatile tool for a wide variety of design problems.

II.5 Using Optimization to Solve Constrained-Variable and Minimax Problems

The optimization algorithm that FARANT employs is taken from Davidon [2] and is one which minimizes a function of several variables. Because the function and the variables are completely described by the user, however, it is possible to use this optimization routine fairly effectively to solve more complicated classes of problems, for example, constrained-variable and minimax problems.

To constrain a variable during the optimization process, one must equate it to a function of another variable which in turn is allowed to vary freely between positive and negative infinity. The function's range should be chosen to correspond to the desired constraint. The argument of that function is passed to Davidon's algorithm, but the user takes the function as his new variable. Examples of this can be seen in the demonstration run in section I.5. The chosen function should be smooth and non-periodic, for example:

Constraint	Suggested Function	Function to Avoid
$x' > 0$	$x' = (x)^2$ or $x' = e^x$	$x' = \|x\|$
$a < x' < b$	$x' = \frac{b-a}{180} \arctan(x) + \frac{b+a}{2}$	$x' = \frac{(b-a)}{2} \sin(x) + \frac{b+a}{2}$

By employing functions such as the above, it is possible to constrain variables to specified ranges and still use the optimization capability of FARANT.

Problems requiring more general constraints on the variables can approximately be solved by constructing an objective function which gets very large as the variables deviate from those required to satisfy the constraints exactly. If the variables are represented by X(*), then a constraint of the form $g(X(*)) = 0$ might be incorporated into the optimization process by adding a term of the form $Q \cdot g^2$, where Q is a large positive constant. Similarly, if one wishes to satisfy $h(X(*)) > 0$, he might add a term of the form $\exp(-R \cdot h)$, where R is a large positive constant. In either of these cases the constraint may not be satisfied exactly, but the optimization process will tend toward the X(*) values which more nearly do satisfy it.

Minimax problems constitute another important kind of optimization in which the maximum value of a function over a limited domain of a given variable is to

be minimized. Minimax problems are sometimes called "equal ripple" problems and are common in band-pass circuits and filters. Since Davidon's algorithm cannot handle minimax problems directly, it is necessary to translate this type of prob into one that it can handle, namely minimization of a given objective function. This can be done as follows:

1. Sample the function at a sufficient number of points along the domain of the given variable, e.g., frequency. This number might be chosen as 5 to 10 times the number of peaks one expects in the function when made equal ripple.
2. Set the objective function equal to a sum of positive terms, one for each sample point, such that the larger points strongly dominate the sum. For instance, these terms might be exponentials of the form $\exp(Q \cdot f_n)$ where f_n is a sample point.
3. Run optimization as usual using Fvalue set equal to this objective function.

If the objective function is truly dominated by the largest terms, and enough samples were taken, then the peaks will be reduced until they are nearly equal. The peaks will subsequently be reduced further until an approximate minimax solution is obtained. It should be noted that this method of solving minimax problems is indirect and approximate and may take considerably longer to obtain a solution than algorithms specifically designed for this purpose. However, an optimization carefully set up in this way may be satisfactory for many problems.

III. FARANT ON THE VAX-11 and the HP 9845A

III.1 Using the Apple Computer as a VAX Terminal

A program has been written [3] which allows the Apple II computer to be used as a remote terminal for the VAX-11. This program acts as a liaison between the Apple keyboard and the VAX and can direct information either to the CRT or hardcopy printer connected to the Apple. In addition to an Apple mainframe, disc-drive, video monitor, and printer, a Micromodem II interface card [4] connec to a Microcoupler, which in turn connects directly to a telephone modular jack, is required for remote access to the VAX.

The "terminal mode" that this program provides is entered by typing RUN VAX followed by a carriage return. The Micromodem II dials the phone number of the Charlottesville VAX (without the use of a telephone handset) and establishes communication. The user must then type his name and valid password to log onto the VAX. In this terminal mode certain characters of the Apple II keyboard are intercepted and either modified or acted upon instead of being sent directly to the VAX. These special characters are the following:

Character	Meaning in Terminal Mode
ctrl-B	\ (backslash)
ctrl-N	[(left bracket)
ctrl-V	_ (underscore)
← (left arrow)	DEL (delete last character)
ctrl-P	toggle the printer on or off
ctrl-A	return to Apple II BASIC (from which GOTO 1000 re-instates terminal mode)

ctrl-X	hang up the phone and return to Apple II BASIC	
ctrl-I*	TAB	(space 8 characters)
shift-M*	]	(right bracket)
→ (right arrow)*	ctrl-U	(ignore entire line)

*These are not actually changed by the terminal simulator program but appear to be changed since their meanings are not obvious from the keyboard.

While in terminal mode the user can access and run FARANT files through the VAX operating system. It is proposed as a future project to interface Apple II graphics capabilities with the VAX version of FARANT, but as of this date, graphics are only available with the HP 9845A version of FARANT, not with the VAX version.

The terminal mode of the Apple also requires a "switch" to be set in the VAX in order that all characters can be delivered to the printer. When prompted for commands from the VAX monitor, i.e., the dollar-sign prompt $, one can type SET TER/CRF=1 to set that switch to the necessary value.

III.2 Peculiarities of VAX BASIC

The implementation of FARANT on the VAX required several minor modifications to the HP 9845A version of the program. The modifications reflect small differenc and quirks of individual BASIC compilers and while the impact on the user is minimal in many cases, it is important in others. The following list points out most, though by no means all, of the compatability problems that were taken care o

HP 9845A BASIC	VAX-11 BASIC version 1.0
OPTION BASE 1 is needed to set the lower limit of array dimensions to 1 rather than 0.	All arrays have lower bounds = 0 but MAT statements ignore the 0th row and column.
Arrays are passed to subroutines with an asterisk identifier, e.g., A(*).	Arrays are passed to subroutines with a comma identifier, e.g., A(,).
PRINT (with or without a format) A(*) is allowed when printing arrays.	Only MAT PRINT A (unformatted) or PRINT A(i,j) (with or without a format) is allowed.
Scalar multiplies or divides are allowed, e.g., MAT A=B/(2), MAT A=(10)*B.	Only scalar multiplies are allowed, e.g., MAT A=(10)*B, and the order must be as shown.
MAT A = INV(A) is allowed.	MAT A = INV(A) is not allowed.
The determinant function DET can operate on an array argument or return the determinant of the last array inverted.	The DET function cannot take an argument. It always returns the value of the last array inverted.
Zero-determinant arrays are inverted in an unpredictable way, but they do not cause execution errors. Post-inversion checking of DET is a valid error handler.	Inversion of a zero-determinant array causes an execution error and must be handled by other means. Mtrans uses the ON ERROR GOTO / RESUME structure.
The REDIM statement is allowed and preserves all values in the array, even when shrinking the working array size.	Matrices can only be redimensioned implicitly and only when they are redefined, e.g., MAT A = ZER(2,3). FARANT uses SUB COPYMAT to copy 4 X 4 arrays.

HP 9845A BASIC	VAX-11 BASIC version 1.0
Any variable can be passed by reference between subroutines.	Individual array elements cannot be passed by reference out of a subroutine if the entire array was passed to that subroutine. The updated BASIC compiler may have fixed this.
Memory is scarce so the data-base is kept in short precision and dimensioned (51,18).	Memory is plentiful so the data-base is kept to full precision and dimensioned (101,18).
Multiple assignments are allowed, e.g., A = B = 0. Multiple statements are not allowed.	Multiple assignments are interpreted as logical equations, i.e., wrong. Multiple statements are allowed, e.g., A = 0 \ B = 0.
The logical operators yield +1 if true, 0 if false, e.g., (8 > 5) evaluates to +1.	Logical operators yield -1 if true, 0 if false.
Trigonometric functions can be evaluated in DEGrees, or RADians.	All trigonometric functions are in radians.
Base 10 logarithms are obtained with LGT(n).	Base 10 logarithms are obtained with LOG10(n).
Variable names in SPACE DEPENDENT mode have only their first letter capitalized. Keywords are distinguished by using all capital letters.	Variable names use all capital letters Some of the unexpected keywords are: ERR FF GE GT IMP LEN LF NUM REF SI SO SP TYP VAL.

HP 9845A BASIC	VAX-11 BASIC version 1.0
FARANT variables Count, Dat, Del, Place$, Type$ are VAX BASIC keywords.	DBNUM%, DB, DL, PLACEMENT$, SERPAR$ are used in the VAX version of FARANT.
Statements can have labels in addition to line numbers and GOTO's can reference either, e.g., GOTO Dotrans or GOTO 1200.	Statement labels are not allowed and all GOTO's must use constant: e.g., GOTO 1200.
Comments denoted with exclamation points are allowed anywhere.	Comments are not allowed in DATA statements, nor can they contain additional exclamation points.
READ statements don't require DATA statements until they are executed.	READ statements will not compile without at least one DATA statement.
Integer constants do not appreciably speed up loops and slow array element references.	Integer loops, e.g., FOR I% = 1% TO 10% are much faster.
The LIN(n), SPA(n), TAB(n) functions are all allowed in unformatted PRINT statements.	Only TAB(n) is allowed.
Formatting syntax is quite different from VAX BASIC's formatting.	A blatant formatting error exists in version 1.0 BASIC: PRINT USING "##.###";.5421E-19 yields .054, not .000

RENumbering program lines is accomplished by a single statement, e.g., REN 100,5.

All BASIC statements are automatically stored in order by their line numbers.

To renumber a program, one must use a Library Resequencing routine whic takes many statements.

In SOS, BASIC source code requires an extra line number which does not automatically order the program lines until execution.

III.3 FARANT Program Pieces on Each Machine

Two aspects of the VAX make it necessary to store FARANT in a slightly different way than is done on the 9845:

(1) The VAX makes available to the user enough storage space that all of FARANT fits into the VAX easily, even if the user has written long programs to go with it. The 9845's 64K of memory is nearly full when it contains the text of FARANT alone.

(2) VAX BASIC is compiled separately from execution whereas HP 9845 BASIC is compiled step by step as it is executed. Therefore, since compilatio of the body of FARANT takes over a minute on the VAX, it is convenient to keep on file a compiled version of the part of FARANT which the user does not have to change.

The order of FARANT's subroutines is listed in the summary of section I.4 and will be used to describe the program pieces stored on each machine. Note that at the very beginning of FARANT is kept a three line "main program," i.e., not a subroutine, that dimensions the common storage space, calls SUB Farstart, and ends. Thus, for all practical purposes, FARANT can be thought of as beginning at SUB Farstart and consisting of nothing but subroutines.

The VAX has the following files stored on the default device DBAØ with directory name [FENSTER]:

Name.type;version	Description
FARANT.BAS;1	is the entire BASIC text of FARANT which includes all subroutines except Smith and Splot.
FARSUB.BAS;1	is the portion of FARANT that the user can leave as is, i.e., all subroutines (except Smith and Splot) from the beginning through SUB Optimize.
FARSUB.OBJ;1	is the compiled version of FARSUB.BAS;1 created with the statement: $ BASIC/DOUBLE/WORD FARSUB.BAS;1
FARAFT.BAS;1	is the user-portion of FARANT which comes after the other subroutines and consists of SUBs Nread, Pread, Farstart, and Cktanalysis, i.e., FARSUB.BAS;1 + FARAFT.BAS;1 ≡ FARANT.BAS;1
RUNFAR.COM;1	is a command-language file that is used to link the user portion of FARANT to the compiled subroutines and run them both.

FARANT files on the HP 9845A cassette tape labelled "FARANT" are the following:

Name	Type	Description
FARANT	ALL	is the entire BASIC text of FARANT including all subroutines plus the HP 9845 typing-aid key definitions.
FSHORT	ALL	is the entire BASIC text of FARANT except 4 seldom used subroutines: Lossyline, Flip, Nread, Pread. It also contains the typing aids.
LOSSYL FLIP NREAD PREAD	DATA DATA DATA DATA	These are the subroutines stored in separate files that were deleted from FARANT to make FSHORT. Each contains one entire subroutine from the SUB to the SUBEND statement.
FARAFT	DATA	is identical to the VAX's FARAFT.BAS;1 and contains SUBs Nread, Pread, Farstart, and Cktanalysis.
FARKEY	KEYS	contains just the typing-aid key definitions. These definitio are included in FARANT and FSHORT above.

III.4 File Manipulations

FARANT files stored on cassette for the HP 9845A are moved to and from memory with simple statements. Since the full version of the program uses nearly all of memory, however, it sometimes becomes necessary to delete subroutines

to make room for user programs and variable storage. Subroutines that can be deleted without affecting the rest of FARANT are: Rlc, Source, Trline, Tf, Lossyline, Flip, Branch, Nload, Nperformance, Smith, Splot, Gammaz, Nread, and Pread. These SUBs are not called by any other subroutines unless the user calls them from Cktanalysis. When a subroutine is added or deleted, it should be handled as a complete unit, i.e., from the SUB to the SUBEND statement. FSHORT, for instance, is missing four SUBs and has room for about two full pages of user program statements. The following statements show useful ways of handling various program pieces on the 9845:

LOADALL "FARANT"	With the tape in the right-hand reader, this brings all of FARANT into memory. Parts must be deleted before running it, however.
LOADALL "FSHORT"	This is recommended to get started.
GET "LOSSYL",8000	SUB Lossyline is added to memory starting at line 8000, using the line number increments that were saved, and higher numbered lines are deleted.
LOAD KEY "FARKEY"	The typing-aid key definitions are put into memory.

EDIT LINE 6100	Allows the user to view and alter lines around 6100, where SUB Cktanalysis has room reserved for him to store statements.
STOREALL "MYPROG"	All of memory is placed on tape with the name "MYPROG". Names must be $\leq$ six characters.
DEL 4625,4755	This deletes SUB Gammaz, for example. (Line numbers should be checked before deleting SUBs.)
SAVE "MYCKT3",6000,9999	Saves SUB Farstart and Cktanalysis where the user typically has his program text.
GET "MYCKT2",6000	Adds the contents of "MYCKT2" to memory beginning at line 6000.

On the VAX, each user with a valid account has a directory name under which files can be stored. FARANT program files are stored under the directory name [FENSTER] but can be copied to one's own directory with the COPY command. When prompted with the dollar sign prompt from the monitor, one would type:

```
$ COPY [FENSTER]*.*;1  *.*;1
```

This statement copies the [FENSTER] version of the FARANT files (whose version numbers are all 1) to one's own directory. All [FENSTER] FARANT files are protected from user editing or deleting but can be copied to another user's directory with this statement and can then be edited, listed, renamed and so forth.

BASIC programs on the VAX can be manipulated from either the "BASIC environment" or from the Digital Command Language monitor DCL, whose prompt is the dollar sign. The author strongly recommends the use of the latter class of operation, despite the seeming complexity of the DCL language and the various modes one can enter from DCL.

In the DCL environment, programs must separately be compiled, linked and executed. FARANT, however, can be run with one statement by using the command-language file RUNFAR.COM;1 which compiles the user section of FARANT, links it to the already compiled version of the rest of FARANT, and runs the resulting executable image. Thus, after one has stored all his FARANT commands in FARAFT.BAS;1, he can run FARANT with the following statement:

```
$ @RUNFAR FARAFT
```

If one used MYCKT3 instead of FARAFT in the statement above, the file MYCKT3.BAS where n was the highest version number of MYCKT3.BAS files, would be compiled, linked with the rest of FARANT, and run. Thus, one can edit FARAFT.BAS;1 to contain commands that control FARANT, save it with a name like MYCKT3.BAS, and run it using the @RUNFAR statement. The contents of RUNFAR.COM;1 are:

```
100     $BASIC/DOUBLE/WORD 'P1'
200     $LINK/EXECUTE='P1' FARSUB + 'P1'
```

```
300          $DELETE 'P1'.OBJ.*
400          $RUN 'P1'
```

where 'P1' stands for the name of the file in the @RUNFAR _____ statement. File type .BAS is assumed for 'P1'.

Since the LINK operation is a complex one, it is not unusual for the @RUNFAR statement to take 30 seconds or more, even if the user has no program statements at all in SUB Cktanalysis. As long as no error messages are generated, this period of inactivity is probably due to compilation and linking time and should not be interrupted.

III.5 Typing Aids on the HP 9845A

The typing-aid keys located in the upper right corner of the HP 9845A keyboard area can be defined by the user to stand for various combinations of keystrokes. A particularly useful set of typing aids for FARANT has been stored with the FARANT and FSHORT files and is also stored by itself in the file called FARKEY. There are 16 actual keys numbered 0 to 15, and used in combination with the shift key, they form 16 more which are numbered 16 to 31. Some keys are left undefined, and the user can define these or any other keys to his liking. FARANT defines the following keys:

```
0:   CALL Rlc( (*)," ",   ,   ,   ," ",   )
1:   CALL Trline( (*),   ,   ,   )
2:   CALL Cas( (*), (*))
3:   CALL Par( (*), (*))
4:   CALL Ser( (*), (*))
```

```
 6:   REWIND":T14"  Execute
 7:   REWIND":T15"  Execute
 8:   PRINTALL IS 0  Execute     (all typing is printed to paper)
 9:   PRINTER IS 0  Execute      (all PRINT statements go to paper)
10:   DUMP GRAPHICS  Execute     (graphics printed to paper)
11:   GRAPHICS  Execute          (displays the graphics memory)
12:   EDIT
13:   EDIT LINE
14:   LIST
15:   SCRATCH
22:   CAT":T14"                  (a catalog of files is printed for
                                  left-hand tape)
23:   CAT                        (same for the right-hand tape)
24:   PRINTALL IS 16  Execute    (all typing goes only to CRT)
25:   PRINTER IS 16  Execute     (PRINT statements go only to CRT)
26:   GCLEAR  Execute            (erases the graphics memory)
27:   LETTER  Execute            (allows labelling of graphics)
28:   LINE TYPE 1  Execute       (can use before KEY 27)
30:   LIST 9999  Execute         (displays AVAILABLE MEMORY)
```

A plastic overlay clearly labels the function of the various keys that FARANT defines for the user.

III.6 A Word on VAX Text Editing

One of the powerful features of the VAX is its text editor SOS which the author recommends over the BASIC environment. The use of SOS is described in

great detail in the "VAX-11 Text Editing Reference Manual." BASIC syntax rules and language elements are fully documented in the "VAX-11 BASIC User's Guide" and the "VAX-11 BASIC Language Reference Manual," both of which are good references but need not be studied in order to use FARANT. (For the user who wants an overview of the VAX's operating system as well, the "VAX/VMS Primer" is succinct and quite helpful.) The text editor itself is accessed by the $ EDIT filename.type statement, at which point one receives the asterisk prompt instead of the dollar sign. A comprehensive summary of Edit-Mode commands is given on page A-2 of the Text Editing reference manual. Some of the most useful of these commands for editing FARANT files are the following:

Alter	Print
Delete	Replace
End	Substitute
Find	Save World
Help	eXtend
Input	Ret and Esc

Within Alter mode, for instance, commands are analogous to the text editing features of the HP 9845. Alter-mode is summarized on page 2-28. The transfer of control between all the text editing modes and DCL level is well diagrammed on page 2-5.

In addition to knowledge of the text editor, it is quite useful to understand several control characters that the VAX responds to:

ctrl-Y is a general interrupt "yoohoo" that returns the user to DCL level. It should not be used, for instance, in Edit-mode until the current file is stored on disk, but is useful to stop long listings, unwanted computations, etc. from DCL.

ctrl-S "stops" the flow of characters sent by the VAX to the termin in any mode of operation, from which point:

ctrl-Q resumes the "queing" of characters from the exact point it was stopped, or

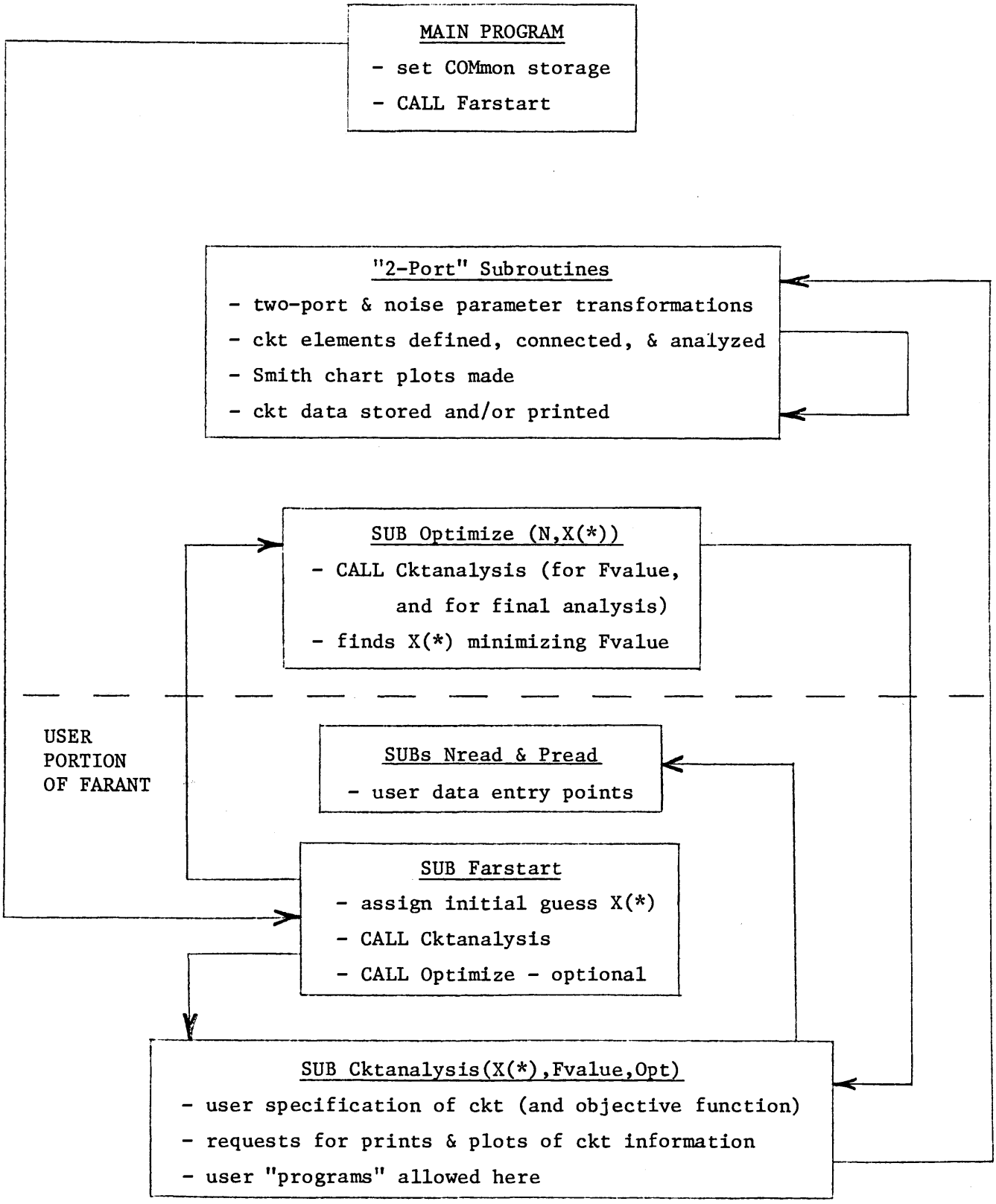
ctrl-O ignores the remaining "output" and returns to the current mode, e.g., DCL or the text editor.

ctrl-U "undoes" the current line that the user had been typing, in order that he may start fresh.

These control characters give the user a great deal of power no matter what mode of operation he may be in. They should be used with discretion, however, and not struck several times in succession. The VAX maintains a "type-ahead" buffer which menas that all characters typed at the terminal are heard by the VAX and acted on in the order that they were typed in -- even if they are not echoed back to the user's CRT for some time! This feature allows the user to type his commands at any rate he desires but can cause mysterious results if it is not understood.

IV. FROM THE PROGRAM'S POINT OF VIEW

IV.1 Flow of Control in FARANT



IV.2 Initializations in SUB Cktanalysis

OPTION BASE 1

This sets the default for the lower limit of dimensioned arrays to 1 rather than 0. Thus, DIM E(6,4) creates an array whose rows are 1 to 6 and whose columns are 1 to 4. This statement is needed in every subroutine on the 9845 as well. It is not needed on the VAX.

COM Nogo, Zo, F, Count, SHORT Dat(51,18) (--full precision DB(101,18) on the

This statement sets up what is known as common storage space for these variables such that all subroutines with a similar COM statement can use them. (On the HP 9845 they remain assigned even after a STOP, or when the user RUNs his program again, unless they are explicitly re-assigned or a SCRATCH C is executed.)

Nogo - a flag set to 1 whenever a requested type of two-port parameters cannot be obtained by SUB Mtrans.

Zo - the variable holding the reference impedance for S-parameters. It is set = 50 at line 6090 of SUB Cktanalysis.

F - the variable holding frequency. The user must assign his frequencies of analysis to this variable.

Count - holds the number of rows of the data-base matrix that have been assigned by SUB Saveckt in the current run. It is set = 0 at line 6085 of SUB Cktanalysis and incremented only by SUB Saveckt.

SHORT Dat(51,18) - This is the data-base matrix which holds up to 50 rows of circuit data. SHORT precision retains about 6 significant digits and allows an exponent in the range $\pm$ 63.

DIM A(6,4),B(6,4),...,H(6,4)

These matrices are used to hold two-port descriptions. The size is (6 X 4) to allow for a partitioned (4 X 4) of 4 complex two-port parameter

(see section IV.3), the labels Pset and Nset in elements (5,1) and (5,2), and 4 noise numbers in row 6. Note that BASIC distinguishes between F (used for frequency) and F(*) (a two-port matrix identifier).

FIXED 3 (--not used on the VAX.)

The default for unformatted printout is set to 3 digits to the right of the decimal point.

DEG (--not used on the VAX.)

The default for trigonometric functions is set to degrees. Subroutines which use trig functions need the DEG statement also.

IV.3 Complex Number Manipulations and Numerical Accuracy

The circuit analysis routines in FARANT use complex numbers for many of their calculations. Voltages and currents are considered to be "phasors" of some magnitude and phase angle at a fixed frequency. Immittances, reflection coefficien noise correlation coefficients, etc. are all complex numbers and are manipulated using both their real and imaginary parts. The Euler identity for expanding an imaginary exponential is used extensively:

$$Ae^{j\theta} = A[\cos(\theta) + j \sin(\theta)]$$

Alternatively, any complex number can be interpreted as a phasor having a magnitude and an angle:

$$a + jb = Mag \cdot e^{j\theta}$$

Complex algebra can effectively be handled with matrix algebra by using the following mapping:

$$z_1 = a + jb \leftrightarrow \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

$$\text{Mag } (a + jb) \leftrightarrow \left\{ \text{Det} \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \right\} \frac{1}{2}$$

$z_1 \; z_2 \leftrightarrow$ matrix product (order not important)

$$\frac{1}{z_1} \leftrightarrow \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1}$$

(2 X 2) matrix of complex z's $\leftrightarrow$ (4 X 4) (partitioned) matrix of real numbers

Product of complex (2 X 2)'s $\leftrightarrow$ product of their (4 X 4)'s

Squared magnitude of a complex (2 X 2)'s determinant $\leftrightarrow$ determinant of its (4 X 4)

The correct phase angle of a complex number is obtained from the principal-value arctangent function by an appropriate offset of $\pm$ 180°. As a utility to FARANT, the following subroutine performs the transformation of complex numbers from rectangular to polar coordinates:

SUB Polar(X,Y)

X,Y are input as rectangular coordinates of a complex number and output as polar coordinates (magnitude, phase) with phase angle θ such that $-180° < \theta \leq 180°$

IF X*Y = 0 THEN θ = [SGN(X)(SGN(X)-1) + SGN(Y)]*90

IF X*Y <> 0 THEN $\theta = \text{TAN}^{-1}(Y/X) - [\text{SGN}(X)-1]*\text{SGN}(Y)*90$

$$\text{Mag} = (X^2 + Y^2)^{\frac{1}{2}}$$

The use of (4 X 4) matrices of real numbers to represent complex (2 X 2)'s has certain drawbacks which have only partially been overcome by FARANT's software.

One problem concerns the speed of matrix operations. The (4 X 4) matrix requires twice the number of adds and multiplies to perform a matrix multiply or matrix inverse as would the actual complex (2 X 2). Since the HP 9845's "MAT" statements are somewhat faster than an equivalent number of arithmetic operations, this factor of 2 reduction in speed is really no more than a factor of about 1.5, and the ease of programming using the MAT statements partially compensates for this slower rate of arithmetic processing. On the VAX, however, the situation is much more severe. MAT statements executed in VAX BASIC take between 20 and 100 times more time than a corresponding number of simple arithmetic operations. Thus, in order to gain the full speed of the VAX hardware, much of FARANT will have to be re-programmed to avoid using MAT statements (and compiled with the /NOSETUP qualifier). The (4 X 4) representation of complex (2 X 2)'s should probably be reduced to a (2 X 4) storage format at the same time that this re-programming is done.

Another drawback of (4 X 4) matrices is the need to keep the complex number representation precise, i.e., to prevent roundoff errors from making the diagonal elements of a single complex number unequal. For instance, each (2 X 2) portion of any (4 X 4) should hold the values {a, b, -b, a} for a complex number z = a + jb. To overcome the roundoff de-symmetrization which occurs after a matrix multiplication or inversion, these (2 X 2) portions must be made to have the above form, say, by averaging their diagonal elements. This averaging operation is done in SUB Mtrans, SUB Cas, and several other places where (4 X 4)'s are inverted or multiplied.

Inverting (4 X 4) matrices of this sort also poses the problem of numerical accuracy when the determinant of that matrix is very small. It was found that

a quick check of the relative size of the determinant was necessary to determine if the inversion was losing an unacceptable number of digits. The DET function is available in BASIC; when applied to a (4 X 4) that represents a complex (2 X 2), its value is the squared magnitude of the complex determinant of the (2 X 2). The value of DET was used in an error analysis of this type of (4 X 4) matrix inversion [5] to arrive at the following comparison:

$$\text{If } \text{DET}[Z] \leq 10^{-n} \left(|Z_{11}|\cdot|Z_{22}| + |Z_{12}|\cdot|Z_{21}|\right)^2$$

$$\text{Then about } \frac{n}{2} + 1 \text{ digits are lost in performing } [Z]^{-1}.$$

Instead of simply checking for a zero determinant, this comparison is used with n = 8 for the HP 9845 (or n = 12 for the VAX in double precision) to check if the determinant is near enough to zero to warrant the taking of precautionary steps before performing the inversion. SUB Mtrans uses this comparison, for example, to decide if it can obtain with sufficient accuracy a given set of two-port parameters that requires a matrix inversion. In making this comparison, the magnitudes of Z_{ij} are estimated as the sum of the absolute values of their real and imaginary parts.

Thus, in summary, the accuracy of manipulating complex numbers in this way is maintained to a very high level by keeping the diagonal elements of single complex numbers equal and by preventing matrices with very small determinants from being inverted. Arithmetic speed could be improved by about 30 to 50% by circumventing the matrix representation of complex numbers on the HP 9845, but could be enhanced by a factor of 50 to 100 by the appropriate re-programming of the VAX version.

IV.4 Two-Port Parameter Definitions and Transformations

Standard definitions of the two-port parameter sets [1], [6], [7] are used throughout FARANT. These are as shown:

1. ABCD-Parameters

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}$$

2. Z-Parameters

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \times \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

3. Y-Parameters

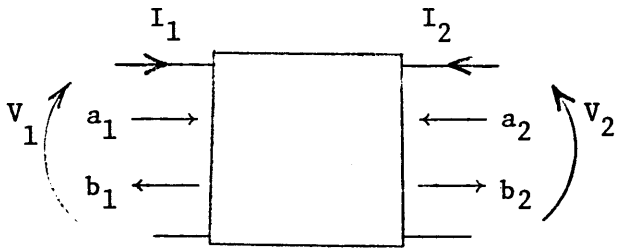
$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

4. S-Parameters

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

5. T-Parameters

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \times \begin{bmatrix} b_2 \\ a_2 \end{bmatrix}$$



$$a_1 = \frac{V_1 + I_1 Z_o}{2\sqrt{Z_o}} \qquad b_1 = \frac{V_1 - I_1 Z_o}{2\sqrt{Z_o}}$$

Transformations from one parameter set to another employ the following formulas:

$$[Z] = \frac{1}{C}\begin{bmatrix} A & AD-BC \\ 1 & D \end{bmatrix} \qquad [A] = \frac{1}{Z_{21}}\begin{bmatrix} Z_{11} & Z_{11}Z_{22}-Z_{12}Z_{21} \\ 1 & Z_{22} \end{bmatrix}$$

$$[Y] = [Z]^{-1} \qquad [Z] = [Y]^{-1}$$

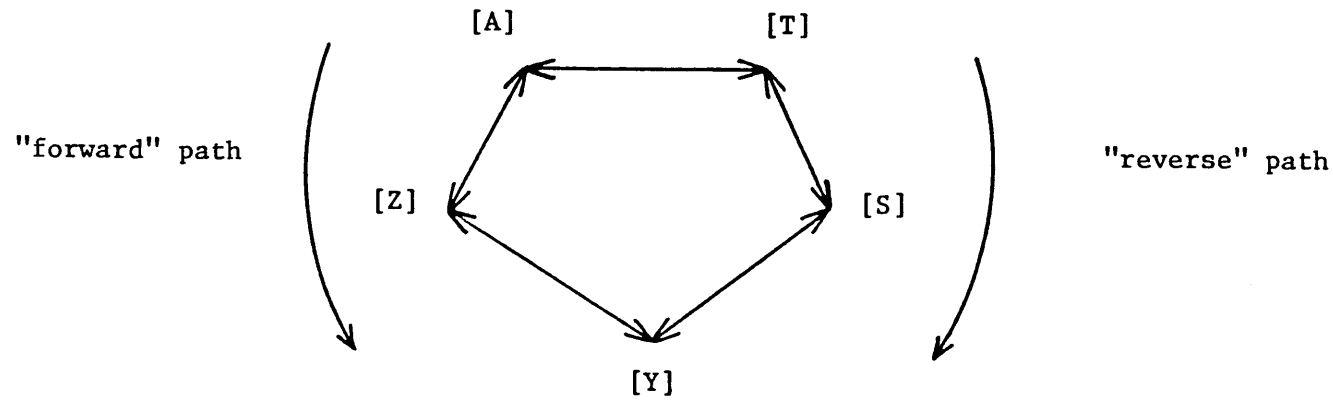
$$[S] = \Big[[1] - [Y]Z_0\Big]\times\Big[[1] + [Y]Z_0\Big]^{-1} \qquad [Y] = \Big[[1] - [S]\Big]\times\Big[[1] + [S]\Big]^{-1}\frac{1}{Z_0}$$

$$[T] = \frac{1}{S_{21}}\begin{bmatrix} 1 & -S_{22} \\ S_{11} & S_{12}S_{21}-S_{11}S_{22} \end{bmatrix} \qquad [S] = \frac{1}{T_{11}}\begin{bmatrix} T_{21} & T_{11}T_{22}-T_{12}T_{21} \\ 1 & -T_{12} \end{bmatrix}$$

$$[A] = \frac{1}{2}\begin{bmatrix} T_{11} + T_{12} + T_{21} + T_{22} & Z_0\,(T_{11} - T_{12} + T_{21} - T_{22}) \\ \frac{1}{Z_0}\,(T_{11} + T_{12} - T_{21} - T_{22}) & T_{11} - T_{12} - T_{21} + T_{22} \end{bmatrix}$$

$$[T] = \frac{1}{2}\begin{bmatrix} A + \frac{B}{Z_0} + CZ_0 + D & A - \frac{B}{Z_0} + CZ_0 - D \\ A + \frac{B}{Z_0} - CZ_0 - D & A - \frac{B}{Z_0} - CZ_0 + D \end{bmatrix}$$

SUB Mtrans can perform the set of cyclical transformations shown here:



This cyclic patth was chosen on the basis of its simplicity (for example, [A] → [Z] and [Z] → [A] are equivalent transformations) and its completeness as far as closing the loop. SUB Mtrans chooses the shortest path to the desired parameter set and usually only performs 1 or 2 transformations. The [A], [Z], [Y] matrices are used for cascade, series and parallel connections, the [S] matri: for plotting and often for printing, and the [T] matrix primarily to provide an alternate path to the desired matrix if one path fails.

Handling Undefined Matrices:

In many cases, a particular matrix will not be defined because its elements are infinite. For example, the [Y] matrix of a parallel impedance, [Z] of a series impedance, [Y] or [Z] of a lossless half-wave transmission line, [A] or [T] of a "broken" or "shorted" 2-port, and even [S] for a few special cases can ideally be infinite. Given a defined matrix and a transformation to be performed, however, SUB Mtrans tests if the result will be finite by checking for division by 0 or the inversion of a matrix whose determinant is practically zero before making the transformation.

Since there exist two possible "paths" to the desired matrix, this cyclical transformation will only fail if:

1. the final matrix is infinite
2. [S] and [Z] don't exist, or
3. [A] (and [T]) and [Y] don't exist (note that [A] and [T] are finite or infinite together)

Now case 2 is actually impossible, and we assume case 3 can't happen for any meaningful 2-port. Thus, the only consideration is what to do when the requested matrix is infinite. Each subroutine that uses Mtrans deals with that locally.

Flow of Control in Mtrans:

Program Line	Matrix Change (P→N)
1_1	1 → 2
1_2	2 → 3
1_3	3 → 4
1_4	4 → 5
1_5	5 → 1
GOTO 1_1 (forward path)	
1_6	5 → 4
1_7	4 → 3
1_8	3 → 2
1_9	2 → 1
1_{10}	1 → 5
GOTO 1_6 (backward path)	

Mtrans first chooses the shortest path by the statement

IF (forward distance: (N>=P)*(N-P) + (N<P)*(5+N-P)) <3 THEN

ON P GOTO $1_1,1_2,1_3,1_4,1_5$

ON 6-P GOTO $1_6,1_7,1_8,1_9,1_{10}$

(P = present type of parameter set)

If a infinite matrix is encountered, the variable Nogo is set equal to 1. This variable is held in common so all other subroutines can check Nogo after calling Mtrans to see if it failed. In all cases X(5,1) gets the type parameters of the final matrix that Mtrans obtained.

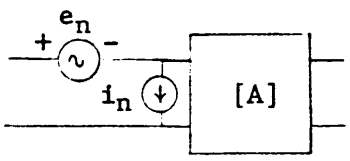
IV.5 Noise Parameter Definitions and Transformations

In general, the various sets of two-port noise parameters consist of four real numbers which depend only upon the two-port and completely describe its noise properties over a narrow frequency band [8]. The process of modelling the noise of a two-port consists of removing the noise waveform from the interior of the circuit and assigning its equivalent RMS spectral components to noise sources placed at the terminals of the two-port. For instance, the open circuit voltages

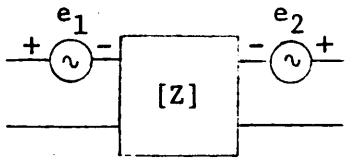
at a given frequency appearing at the terminals of a two-port can be represented by a random voltage generator in series with each port. The four numbers needed to describe this representation are the average squared magnitude of each voltage--two real numbers--and the complex average of their product--one complex or two real numbers. All noise representations are equivalent and can be thought of as derived from this fundamental model of two series voltage generators, or from their dual of two shunt current generators. The following will be a comprehensive summary of the noise representations used in FARANT with emphasis on the transformations employed in SUB Ntrans to manipulate them.

Several of FARANT's noise representations can be described with simple diagrams and of these, four relate directly to two-port parameter sets:

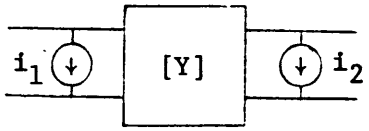
(1)



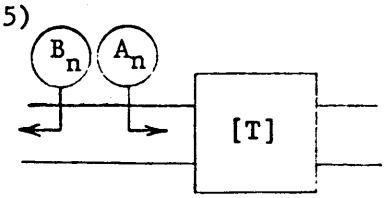
(2)



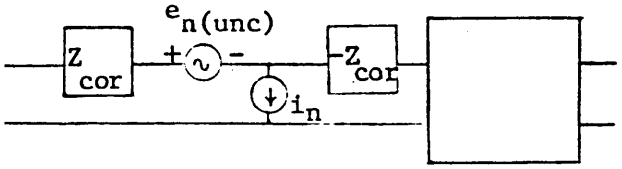
(3)



(5)

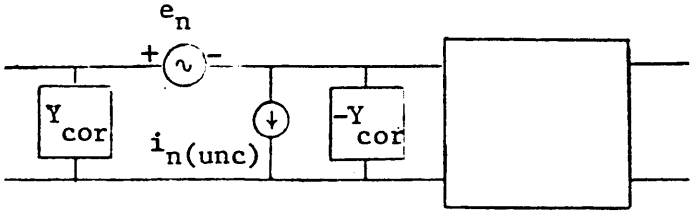


(7)



$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = [T] \begin{bmatrix} b_2 \\ a_2 \end{bmatrix} + \begin{bmatrix} A_n \\ B_n \end{bmatrix}$$

(8)



$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = [A] \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} + \begin{bmatrix} e_n \\ i_n \end{bmatrix} \qquad \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = [Z] \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \qquad \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = [Y] \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

In the following definitions, the variable names in the above diagrams will be used in conjunction with complex conjugates (asterisks), magnitudes (vertical bars), and time averages (horizontal bars) to derive the quantities used for noise parameter representations. Some noise parameters are used in several representations--where the symbol is the same, the meaning is the same, e.g., G_n is used by representation (1), (3), (4) and (7). T_o is the standard temperature of 290°K, k is Boltzmann's constant 1.38×10^{-23} (Watts/ΔHz °K), and Δf is the bandwidth of the noise spectral components.

(1): R_n G_n $|\rho|$ ϕ (see references [8], [9])

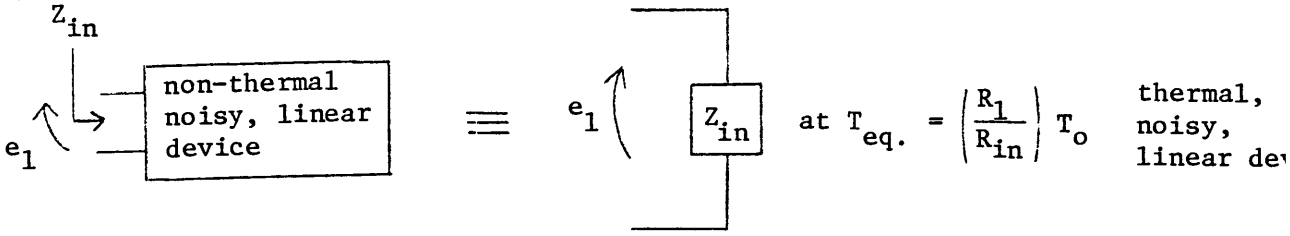
$$R_n = \frac{\overline{|e_n|^2}}{4kT_o\Delta f} \qquad G_n = \frac{\overline{|i_n|^2}}{4kT_o\Delta f} \qquad \rho = \frac{\overline{e_n^* i_n}}{\sqrt{\overline{|e_n|^2}\ \overline{|i_n|^2}}}$$

(2): R_1 R_2 $|\rho_v|$ ϕ_v

$$R_{1,2} = \frac{\overline{|e_{1,2}|^2}}{4kT_o\Delta f} \qquad \rho_v = \frac{\overline{e_1^* e_2}}{\sqrt{\overline{|e_1|^2}\ \overline{|e_2|^2}}}$$

NOTE: R_1, for example, is <u>not</u> an equivalent resistor whose thermal noise in some hypothetical circuit could account for the noise of the device, but rather a number proportional to the mean of a squared voltage over a narrow bandwidth. We can, however, find an equivalent <u>temperature</u>, whose available thermal power

from an actual impedance would be that of the device. This equivalent temperatu
is $\left(\frac{R_1}{R_{in}}\right) T_o$ and for a one-port we have:



available power $P_{av} = \frac{\overline{|e_1|^2}}{4R_{in}}$ for both; $R_1 = \frac{\overline{|e_1|^2}}{4kT_o\Delta f}$

(3): G_1 G_2 $|\rho_c|$ ϕ_c

$$G_{1,2} = \frac{\overline{|i_{1,2}|^2}}{4kT_o\Delta f} \qquad \rho_c = \frac{\overline{i_1^* i_2}}{\sqrt{\overline{|i_1|^2}\ \overline{|i_2|^2}}}$$

NOTE: Instead of the correlation coefficients for representations (1), (2), and (3), FARANT stores the real and imaginary parts of un-normalized quantities which are denoted here as "Cor's". They are defined as follows:

$$Cor = \rho\sqrt{R_nG_n} \qquad Cor_v = \rho_v\sqrt{R_1R_2} \qquad Cor_c = \rho_c\sqrt{G_1G_2}$$

Both the two-port matrices and the data-base store these complex Cor's as the third and fourth noise parameters rather than ρ's. However, the correlation coefficients ρ, ρ_v, ρ_c are still used for input

(in magnitude, phase) to SUBs Nload and Nread and for printout from SUB Pri implied by the conventions in section I.2

(4): T_{min} R_{opt} X_{opt} G_n (see [1], [10])

T_{min} = the minimum equivalent temperature that when added to the temperature of a certain passive source would account for the available noise power at the output terminals of the two-port due to the two-port itself.

$R_{opt} + j\ X_{opt}$ = that impedance which when driving the two-port would minimize the equivalent noise temperature of the two-port to T_{min}.

G_n is defined under (1).

(5): T_a T_b $|\rho_{ab}|$ ϕ_{ab} (see [11], [12])

$$T_a = \frac{\overline{|A_n|^2}}{k\Delta f} \qquad T_b = \frac{\overline{|B_n|^2}}{k\Delta f} \qquad \rho_{ab} = \frac{A_n^* B_n}{\sqrt{\overline{|A_n|^2}\ \overline{|B_n|^2}}}$$

where $$A_n = \frac{-e_n - Z_o i_n}{2\sqrt{Z_o}} \qquad B_n = \frac{e_n - Z_o i_n}{2\sqrt{Z_o}}$$

This noise representation defines equivalent right and left going noise waves A_n and B_n from representation (1)'s noise voltage and current generators e_n and i_n. The available pow

that would be delivered by these generators in a Z_o system is thus split into the two waves and equivalently into the two temperatures T_a and T_b.

(6): T_{min} T_b $|\Gamma_{opt}|$ ϕ_{opt}

The parameters used by this representation are defined under (4) and (5). Γ_{opt} is the optimum source reflection coefficient defined by $R_{opt} + j\, X_{opt}$ and Z_o.

(7): r_n G_n R_{cor} X_{cor} (see [9])

$Z_{cor} = R_{cor} + j\, X_{cor}$ is called the correlation impedance and is the proportionality constant (with dimension of ohms) between the noise current source in the (1) representation and the fraction of the noise voltage source which is correlated to that current. Thus, splitting the noise voltage e_n into an uncorrelated part and another part totally correlate to the noise current i_n, we define Z_{cor} by:

$$e_n = e_{unc} + Z_{cor} i_n \qquad \text{where} \quad \overline{e_{unc} i_n} = 0$$

$$r_n = \frac{\overline{|e_{unc}|^2}}{4kT_o\Delta f} = R_n(1 - |\rho|^2)$$

(8): R_n G_n G_{cor} B_{cor}

This is the dual of (7) and defines $Y_{cor} = G_{cor} + j\ B_{cor}$ by:

$$i_n = i_{unc} + Y_{cor} e_n$$

$$g_n = \frac{\overline{|i_{unc}|^2}}{4kT_o\Delta f} = G_n(1-|\rho|^2)$$

NOTE: G_{cor} and B_{cor} (as well as G_n and g_n) have the dimension of conductance and therefore take on the units of milli-mhos for input and printing, mhos for storage.

SUB Ntrans transforms any of these 8 noise representations into any other by means of the algebra that will follow. The transformations are performed usi the (1) representation as an intermediate step (with the exception of the (5) ↔ (6) transformation) instead of going straight from one representation to another. Note that for all but the transformations involving the (2) and (3) noise parameters, no <u>two-port</u> parameters are required, i.e., noise representatio which describe generators only to one side of the two-port can stand alone and do not need the two-port parameters to be transformed into each other. Since Ntrans makes few assumptions about the validity of the noise representation it receives, there exist possibilities of transformations for which the formulas are undefined, e.g., division by zero or taking the square root of a negative number. These situations can occur if a noise immittance or temperature which is defined as a positive quantity, e.g., R_n, G_1, T_a, R_{opt}, etc., is negative on

input to Ntrans, or a correlation coefficient is bigger than 1. Appropriate safeguarding steps are taken whenever such a situation occurs and these are clearly described under each noise transformation.

(2) $\leftrightarrow$ (1) $\leftrightarrow$ (3) Noise Transformations

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} 1 & -Z_{11} \\ 0 & -Z_{21} \end{bmatrix} \begin{bmatrix} e_n \\ i_n \end{bmatrix} \qquad \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} -Y_{11} & 1 \\ -Y_{21} & 0 \end{bmatrix} \begin{bmatrix} e_n \\ i_n \end{bmatrix}$$

(1) $\rightarrow$ (2)

$$R_1 = |Z_{11}|^2 G_n + R_n - 2\mathrm{Re}\{Z_{11}\mathrm{Cor}\}$$

$$R_2 = |Z_{21}|^2 G_n$$

$$\mathrm{Cor}_v = Z_{21}(Z_{11}^* G_n - \mathrm{Cor})$$

(2) $\rightarrow$ (1)

$$R_n = (|Z_{21}|^2 R_1 + |Z_{11}|^2 R_2 - 2\mathrm{Re}\{Z_{11}^* Z_{21} \mathrm{Cor}_v^*\})/|Z_{21}|^2$$

$$G_n = R_2/|Z_{21}|^2$$

$$\mathrm{Cor} = (Z_{11}^* R_2 - Z_{21}^* \mathrm{Cor}_v)/|Z_{21}|^2$$

(1) $\rightarrow$ (3)

$$G_1 = |Y_{11}|^2 R_n + G_n - 2\mathrm{Re}\{Y_{11}\mathrm{Cor}^*\}$$

$$G_2 = |Y_{21}|^2 R_n$$

$$\mathrm{Cor}_c = Y_{21}(Y_{11}^* R_n - \mathrm{Cor}^*)$$

(3) $\rightarrow$ (1)

$$R_n = G_2/|Y_{21}|^2$$

$$G_n = (|Y_{21}|^2 G_1 + |Y_{11}|^2 G_2 - 2\mathrm{Re}\{Y_{11}^* Y_{21} \mathrm{Cor}_c^*\})/|Y_{21}|^2$$

$$\mathrm{Cor} = (Y_{11} G_2 - Y_{21} \mathrm{Cor}_c^*)/|Y_{21}|^2$$

NOTES: (1) → (2) and (1) → (3) are valid provided [Z] and [Y] respectively exist. If these two-port parameters are underdefined, so are the respective noise representations, so Ntrans prints a message to this effect and quits without obtaining the new noise parameters.

For the (2) → (1) and (3) → (1) transformations, the respective [Z] or [Y] has to exist if the initial noise representations were to be meaningful. If [Z] or [Y] do not exist, an error is printed and Ntrans quits. If Z_{21} or Y_{21} is zero, Ntrans sets $R_n = R_1$ or $G_n = G_1$ and sets the other three parameters to zero, but unless R_2 or G_2 were also zero, this action is erroneous and an error message is printed.

(1) → (4)

$G_n = G_n$

$X_{opt} = \mathrm{Im}\{Cor\}/G_n$

$R_{opt} = \sqrt{R_n/G_n - X_{opt}^2}$

$T_{min} = 2T_o(G_nR_{opt} + \mathrm{Re}\{Cor\})$

(4) → (1)

$R_n = G_n(R_{opt}^2 + X_{opt}^2)$

$G_n = G_n$

$\mathrm{Re}\{Cor\} = T_{min}/2T_o - G_nR_{opt}$

$\mathrm{Im}\{Cor\} = G_nX_{opt}$

NOTES: The (1) → (4) transformation is safeguarded in two ways. If $G_n = 0$ the only noise generator is a series voltage source R_n whose $Z_{opt} = \infty$, and $T_{min} = 0$. But since the (4) → (1) transform could not recover the $R_n = 0 \cdot \infty$, Z_{opt} is set to 0 thus setting all noise parameters to 0 if G_n was 0. If a negative quantity appears under the radical, this is equivalent to $|Z_{opt}|^2 < |X_{opt}|^2$ which is invalid, so R_{opt} is set to 0. R_n and G_n should never be negative.

(1) → (5)

$$T_a = \frac{T_o}{Z_o}(R_n + Z_o^2 G_n + 2Z_o \mathrm{Re}\{Cor\})$$

$$T_b = \frac{T_o}{Z_o}(R_n + Z_o^2 G_n - 2Z_o \mathrm{Re}\{Cor\})$$

$$\mathrm{Re}\{\rho_{ab}\} = \frac{T_o}{\sqrt{T_a T_b}\, Z_o}(-R_n + Z_o^2 G_n)$$

$$\mathrm{Im}\{\rho_{ab}\} = \frac{T_o}{\sqrt{T_a T_b}}(2\mathrm{Im}\{Cor\})$$

(5) → (1)

$$R_n = \frac{Z_o}{4T_o}(T_a + T_b - 2\sqrt{T_a T_b}\,\mathrm{Re}\{\rho_{ab}\})$$

$$G_n = \frac{1}{4T_o Z_o}(T_a + T_b + 2\sqrt{T_a T_b}\,\mathrm{Re}\{\rho_{ab}\})$$

$$\mathrm{Re}\{Cor\} = (T_a - T_b)/4T_o$$

$$\mathrm{Im}\{Cor\} = \frac{\sqrt{T_a T_b}}{2T_o}\,\mathrm{Im}\{\rho_{ab}\}$$

NOTES: The (1) → (5) transform stores $|\rho_{ab}|$ and ϕ_{ab}, not the real and imaginary parts. It is safeguarded by setting both of these quantities to 0 if either T_a or $T_b = 0$. Neither temperature should ever be negative

(5) → (6)

$$T_{min} = T_d - T_b$$

$$T_b = T_b$$

$$|\Gamma_{opt}| = \sqrt{T_a T_b}\,|\rho_{ab}|/T_d$$

$$\phi_{opt} = \pi - \phi_{ab}$$

$$T_d = \left(T_a + T_b + \sqrt{(T_a + T_b)^2 - 4T_a T_b |\rho_{ab}|^2}\right)/2$$

(6) → (5)

$$T_a = T_{min} + (T_{min} + T_b)|\Gamma_{opt}|^2$$

$$T_b = T_b$$

$$|\rho_{ab}| = (T_{min} + T_b)|\Gamma_{opt}|/\sqrt{T_a T_b}$$

$$\phi_{ab} = \pi - \phi_{opt}$$

NOTES: Meys' "T_c" = $T_a T_b |\rho_{ab}|$; "ϕ_c" = ϕ_{ab}. If $T_a T_b = 0$ in the (6) → (5) transform, $|\rho_{ab}|$ is set to 0. If $T_d = 0$, $|\Gamma_{opt}|$ is set to 0.

(1) → (7)

$Z_{cor} = Cor^*/G_n$

$G_n = G_n$

$r_n = R_n - G_n(R_{cor}^2 + X_{cor}^2)$

(7) → (1)

$R_n = r_n + G_n(R_{cor}^2 + X_{cor}^2)$

$G_n = G_n$

$Cor = G_n Z_{cor}^*$

(1) → (8)

$Y_{cor} = Cor/R_n$

$R_n = R_n$

$g_n = G_n - R_n(G_{cor}^2 + B_{cor}^2)$

(8) → (1)

$G_n = g_n + R_n(G_{cor}^2 + B_{cor}^2)$

$R_n = R_n$

$Cor = R_n Y_{cor}$

NOTES: In the (7) and (8) noise representations, Rothe & Dahlke reverse th notation of g_n and G_n from what is used here. If $G_n = 0$ in (1) → (7), or $R_n = 0$ in (1) → (8), then Z_{cor} or Y_{cor} are set to 0 respectively.

One further noise transformation that FARANT performs involves the thermal noise of passive lossy networks. It can be shown [13] that given any passive n-port network at temperature T_{amb}, the average squared magnitudes and cross-correlations of the open circuit noise voltages appearing at each port can be expressed quite simply in terms of the n-port Z-parameters:

$$\overline{e_i e_j^*}\,\Delta f = 4kT_{amb}\Delta f\,\frac{Z_{ij} + Z_{ji}^*}{2}$$

The shunt current generators are similarly expressed in terms of the Y-paramete Thus for two-ports whose Z-parameters are defined, the following relationships define the (2) set of noise parameters:

$$R_1 = \frac{T_{amb}}{T_o} \mathrm{Re}\{Z_{11}\}$$

$$R_2 = \frac{T_{amb}}{T_o} \mathrm{Re}\{Z_{22}\}$$

$$Cor_v = \frac{T_{amb}}{T_o} \frac{Z_{21} + Z_{12}^*}{2}$$

Subroutines Rlc, Lossyline and Nload make use of this derivation of the noise parameters when passed a value for the ambient temperature $T_{amb} > 0$.

IV.6 Calculations and Logic in the Subroutines

SUB Mtrans(X(*),Pset)

The formulas used in Mtrans and the cyclic nature of the transformations are carefully described in section IV.4. The logic at each of the 10 transform starting points is as follows:

1. If Pset = the matrix type to be changed, starting at the present line, then the changes have been successful; set Nogo = 0 and finish.
2. If change is impossible AND Nogo = 1 already, then set X(5,1) = P and subexit. (Failure has occurred in both directions.)
3. If change is impossible but Nogo is still 0, then set Nogo = 1 and branch to the other path to try that.
4. Perform the change normally and then set P = [the type parameters just created].

SUB Ntrans(X(*),Nset)

This subroutine assumes that X(*) contains a valid noise representation on input, with its label in X(5,2), and that Nset is 1 to 8. The transformation are sequenced as shown with the numbers following the L naming the noise transformation. For instance, L17 is the line where the (1) → (7) transform begins.

Transforms to the (1) Noise Parameters	Transforms from (1) to the final Nset
L81	L12 (also does (1) → (3))
L71	L14
L65 → SUBEXIT?	L15 → SUBEXIT?
L51	L56
L41	L17
L31 (also does (2) → (1))	L18
IF Nset = 1, THEN SUBEXIT	These all SUBEXIT with the final Nset.

Each transform uses the present noise parameters, which are assigned to variable A, B, C, D for convenience, assigns new parameters to row 6 of X(*) and their label to X(5,2), and branches or subexits according to the above.

SUB Rlc(X(*),Type$,R,L,C,Place$,Tamb)

	Series Type	Parallel Type
Placed in Series	$[A] = \begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$	$[A] = \begin{bmatrix} 1 & 1/Y \\ 0 & 1 \end{bmatrix}$
Placed in Parallel	$[A] = \begin{bmatrix} 1 & 0 \\ 1/Z & 1 \end{bmatrix}$	$[A] = \begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$
	$Z = R+j(\omega L-1/\omega C)$	$Y = 1/R+j(\omega C-1/\omega L)$

At d-c (0 GHz) the subroutine uses 10^{-10} GHz to calculate an open circuit capacitor or shorted inductor. At a frequency where L and C are exactly resonant and R = 0, 10^{-8} ohms of resistance are added to a series L-C in parallel and 10^{-8} mhos of conductance are added to a parallel L-C placed in series. A value of zero passed to SUB Rlc for R, L or C causes it to omit that element in the two-port and ignore the corresponding term in the Z or Y calculation.

If both R and T_{amb} are positive, Nyquist's formulation of the thermal noise parameters is used to assign a noise representation according to the following:

Placed in Series

Series Type	Parallel Type
$G_1 = G_2 = -Cor_c = \frac{T_{amb}}{290} \frac{R}{R^2 + X^2}$	$G_1 = G_2 = -Cor_c = \frac{T_{amb}}{290} G$

Placed in Parallel

Series Type	Parallel Type
$R_1 = R_2 = Cor_v = \frac{T_{amb}}{290} R$	$R_1 = R_2 = Cor_v = \frac{T_{amb}}{290} \frac{G}{G^2 + B^2}$

SUB Trline(X(*),Zg,Length,K)

ABCD-parameters for the lossless line are assigned as follows:

$$[A] = \begin{bmatrix} \cos(\theta) & jZ_g\sin(\theta) \\ j\sin(\theta)/Z_g & \cos(\theta) \end{bmatrix}$$

where $\theta = 2\pi F \frac{Length \sqrt{K}}{\underline{c}}$

$\underline{c}$ = 11.8028527 x 10^9 in/sec = speed of light in vacuum

SUB Lossyline(X(*),Zgo,Length,K,Cattn,Dattn,Fo,Tamb)

The conductor and dielectric attenuations--Cattn,Dattn--in dB/inch are converted to nepers/inch by $\ell n(10)$ nepers = 20 dB and put in Ac,Ad. It can be shown [14] that for relatively good conductors and dielectrics at microwave frequencies, the distributed impedance of the conductor and admittance of the dielectric are given by:

$$L = Z_{go}\sqrt{K}/\underline{c} \quad (\underline{c} = \text{speed of light in free space})$$

$$C = \sqrt{K}/\underline{c}Z_{go}$$

$$R = \sqrt{F/F_o}\ 2Ac\left(Ac + \sqrt{2Ac^2 + \omega_o^2 LC}\right) \ /\omega_0 C$$

$$G = (F/F_0)\left[2Ad\sqrt{Ad^2 + \omega_0^2\ LC}\right]/\omega_0 L$$

From these the series impedance per length in the conductor and the shunt conductance per length in the dielectric are:

$$z = R + j(\omega L + R)$$

$$y = G + j(\omega C)$$

The complex characteristic impedance of the line and the propagation constant are directly related to z and y by:

$$Z_g = \sqrt{z/y} \qquad \gamma = \sqrt{zy}$$

The angle of γ is taken between 0 and 90° (not including 0); the angle of between -45° and 45°. The ABCD-parameters are then assigned as follows:

$$[A] = \begin{bmatrix} \cosh(\gamma Len) & Z_g \sinh(\gamma Len) \\ \frac{1}{Z_g}\sinh(\gamma Len) & \cosh(\gamma Len) \end{bmatrix}$$

where Len = Length, the physical length of the line.

SUB Source(X(*),Control$,Source$,Gain,R1,R2,Delay)

	V Source	C Source
V Controlled	$[Z] = \begin{bmatrix} R_1 & 0 \\ -R_1\mu & R_2 \end{bmatrix}$	$[Z] = \begin{bmatrix} R_1 & 0 \\ -R_1R_2g_m & R_2 \end{bmatrix}$
C Controlled	$[Z] = \begin{bmatrix} R_1 & 0 \\ -r_m & R_2 \end{bmatrix}$	$[Z] = \begin{bmatrix} R_1 & 0 \\ -\alpha R_2 & R_2 \end{bmatrix}$

A non-zero value of Delay changes the Gain term (Z_{21}) into a complex number with a phase factor of $e^{-j2\pi F\tau}$, where τ is the Delay.

SUB Tf(X(*),Turns1,Turns2)

The ABCD-parameters for an ideal transformer are assigned to X(*):

$$[A] = \begin{bmatrix} \frac{turns1}{turns2} & 0 \\ 0 & \frac{turns2}{turns1} \end{bmatrix}$$

SUB Flip(X(*))

X(*) is first transformed to [Z] or [Y] parameters by Mtrans, and its noise, if any, to the corresponding (2) or (3) parameters. The subroutine swaps the input and output noise, and takes the complex conjugate of the correlation. Then the diagonal elements of X(*) are exchanged, i.e., $X_{11} \leftrightarrow X_{22}$ and $X_{12} \leftrightarrow X_{21}$.

SUB Branch(X(*),Type$)

The open circuit Z_{11} of X(*) is used to define the branch element. If $|Z_{11}| = 0$ or the Z-parameters are undefined, X(*) is left as a noiseless null element whose ABCD-parameters form the identity matrix, and a message is printed to this effect. Otherwise, the ABCD-parameters are assigned as in SUB Rlc. Note

that port 2 is left open and is thus inaccessible. If a shorted stub is desired, for instance, then the short should be cascaded on before the element is made into a branch.

The noise parameters of the branch element--shown as primed quantities--are obtained from the (2) noise parameters of the element in X(*)--unprimed quantities--before being made a branch. For a parallel branch the new noise representation consists of noise voltages:

$$R_1' = R_2' = Cor_v' = R_1$$

For a series branch, the new noise representation has noise currents:

$$G_1' = G_2' = -Cor_c' = \frac{R_1}{|Z_{11}|^2}$$

If the Z-parameters of X(*) are undefined or $Z_{11} = 0$, then no noise is assigned to the branch element.

SUB Nload(X(*),Nset,N1,N2,N3,N4)

If Nset = 0 and the thermal noise is to be assigned, Nload assigns it from the [Z] or [Y] of X(*) but prints an error message if the resulting noise parameters do not describe the noise representation of a passive two-port. For Nset = 1 to 8, Nload assigns the 4 noise parameters in program storage units, and in either case assigns their label to X(5,2).

SUB Polar(X,Y)

See description under Complex Number Manipulations in section IV.3.

SUB Cas(X(*),A(*))

The two-port parameters of the cascade are obtained by post-multiplying the [A] matrix -- ABCD parameters -- for the element X(*) by the [A] matrix

for element A(*) and putting the result in element X(*). If either element's [A] matrix is undefined, meaning it has no forward transmission, the cascade is not performed and a message is printed to this effect.

Before cascading the elements, however, the combined noise is calculated in the following way. The noise in X(*) and A(*), if there is any, is put into the (1) noise representation. Then the "noise correlation matrix" of element A(*) shown below is transformed to the plane of element X(*)'s noise and added to that noise by the following matrix operation:

$$[N]_{Tot} = [N]_{X(*)} + [A]_{X(*)}[N]_{A(*)}[A]^{t}_{X(*)}$$

where [N] = the noise correlation matrix of the (1) noise parameters

$$= \begin{bmatrix} \overline{|e_n|^2} & \overline{e_n i_n^*} \\ \overline{e_n^* i_n} & \overline{|i_n|^2} \end{bmatrix} \propto \begin{bmatrix} R_n & Cor^* \\ Cor & G_n \end{bmatrix}$$

$[A]_{X(*)}$ = ABCD-parameters of element X(*)

t indicates the conjugate transpose of the matrix

Tot refers to the total noise resulting from the cascade of X(*) and A(*), i.e., referenced to the left of element X(*)

X(*) and A(*) refer to the respective two-ports.

SUB Par(X(*),A(*))

SUB Ser(X(*),A(*))

In the Par and Ser subroutines the two-port parameters of the composite elements are placed in X(*) and are obtained by summing the [Y] or [Z] matrices

respectively of the individual elements. If either element has an undefined matrix of the type needed for the connection, the connection is not made and a message is printed to inform the user.

If noise parameters exist for either or both two-ports, they are transformed into the respective types--(2) for Series or (3) for Parallel--and added directly as were the two-port parameters. For example:

$$R_1' = \text{sum of } R_1\text{'s} \quad \text{(of elements X(*) and A(*))}$$

$$R_2' = \text{sum of } R_2\text{'s}$$

$$Cor_v' = \text{sum of } Cor_v\text{'s}$$

For paralleling, G_1', G_2', Cor_c' for the combined circuit are calculated by an analogous addition of those parameters from each two-port.

SUB Saveckt(X(*),Pset,Nset,Kfact)

This is the only subroutine which changes the variable Count, and it does so by a positive increment of 1 each time it is called. The two-port and noise parameters requested by Pset and Nset are obtained by calling SUB Mtrans and Ntrans respectively. These requested parameter sets are stored in the next available row of the data-base, and if this is the first row being stored, i.e., Count now is 1, their labels are stored in array elements (51,1) and (51,2) as well. The parameter types of future rows are compared to these labels on subsequent calls of Saveckt to ensure that the data-base holds but one type of two-port and one type of noise parameters. A message is printed if the Pset or Nset parameter is different from the parameter types that were previously stored in the data-base. A message is also printed if the requested Pset parameters are undefined.

Rollett's K-factor [15] is also computed and stored in the data-base, unless Kfact was negative on input. Y-parameters are used in this calculation unless they are undefined, in which case Z-parameters are used. For either [Y] or [Z] the formula is as follows:

$$K = \frac{2\mathrm{Re}\{X_{11}\}\mathrm{Re}\{X_{22}\} - \mathrm{Re}\{X_{12}X_{21}\}}{|X_{12}X_{21}|}$$

where all the X_{ij} used are either Y or Z parameters. The pass-parameter Kfact is given the value of K in full precision, whereas the data-base stores only the first 6 significant figures.

SUB Nperformance(X(*),Gtype,Rs,Xs,Rl,Xl,Gain,Tn)

The noise temperature is calculated provided Rs $\neq$ 0, there is a noise representation in X(*), and Tn is non-negative on input. The noise temperature of two-port X(*) driven from a source impedance of $R_s + jX_s$ is obtained from the (4) set of noise parameters:

$$T_n = T_{min} + \frac{T_o G_n}{R_s}\left[(R_s - R_{opt})^2 + (X_s - X_{opt})^2\right]$$

Both the data-base and the pass-parameter Tn receive the value of the noise temperature; they receive 0 if the calculation is omitted. In the data-base is also stored R_s and X_s, regardless of the other calculations.

The gain is calculated for Gtype = 1 to 4 but omitted for Gtype = 0. The four types of gain are calculated using ABCD-parameters and possibly Z_s, Z_L, Z_{in} and Z_{out}, the latter two of these being obtained from SUB Zio. However, Z_s and Z_L need only be input to Nperformance if they are required for the requested type of gain. Standard definitions [7] for the various gain types are used by

FARANT and are shown in the following. Their functional dependence on the and load impedances, their expressions in terms of ABCD-parameters, and th equivalence to special cases of the transducer gain are also shown:

Transducer Gain -- Gtype = 1

$$G_T = \frac{P_d}{P_{avS}} = f(Z_s,[A],Z_L) = \frac{4|Z_{in}|^2 R_s R_L}{|Z_{in} + Z_s|^2 \, |AZ_L + B|^2}$$

Power Gain -- Gtype = 2

$$G_p = \frac{P_d}{P_i} = f([A],Z_L) = \frac{|Z_{in}|^2 R_L}{R_{in}|AZ_L + B|^2} \equiv G_T\Big|_{Z_s = Z_{in}^*}$$

Available Gain -- Gtype = 3

$$G_A = \frac{P_{av0}}{P_{avS}} = f(Z_s,[A]) = \frac{R_s}{R_{out}|A + CZ_s|^2} \equiv G_T\Big|_{Z_L = Z_{out}^*}$$

Maximum Available Gain -- Gtype = 4

$$G_{max} = \frac{P_{av0}}{P_i} = f([A]) = \frac{1}{|AB - CD|\left(K + \sqrt{K^2 - 1}\right)} \equiv G_T\Bigg|_{\substack{Z_s = Z_{in}^* \\ Z_L = Z_{out}^*}}$$

where:

P_d = power delivered to the load = $|I_2|^2 R_L$

P_i = power absorbed at the input = $|I_1|^2 R_{in}$

P_{avS} = power available from the source = $|e_s|^2/4R_s$

P_{av0} = power available at the output = $|V_2|^2/4R_{out}\Big|_{I_2 = 0}$

The gain in dB = 10 $\log_{10}(G)$ is stored in the data-base and assigned to the pass-parameter Gain as well. If K < 1 and G_{max} is requested, the maximum stable gain obtained by using a K of unity is calculated and stored instead of G_{max}, and a message is printed warning the user that this is taking place.

SUB Prt(Pset,Nset)

If either Pset or Nset is non-zero and different from that used when Saveckt was called, a loop is set up internally to simulate a new run to store the requested parameters. This is done by filling a (6 X 4) matrix with the two-port and noise parameters from each row of the data-base taken one row at a time, and calling Saveckt with that matrix and the appropriate Pset and Nset as the pass-parameters. Then, if Pset and/or Nset were positive, printout is produced with the proper headings and aligned columns. The K-factor, which is included with the two-port parameter printout, is printed as a number in the range -999 to +9999 such that if it exceeds these bounds, the true value is not printed. It remains correctly stored, however. The noise parameter printout includes a title stating the gain-type that was requested, and headings for each column. User-unit are printed throughout, e.g., milli-mhos, degrees, correlation coefficients, etc.

SUB Smith(Xmin,Xmax,Ymin,Ymax)

Circles of constant resistance or reactance are plotted for normalized values of 0, .2, .5, 1, 2, 5, 10. Loops are set up to plot the 7 resistance circles and the 13 reactive portions of circles and use DATA statements to read in the circles to be plotted and their labels. The user could thus create a chart with a different number of plotted circles by editing these DATA statements and the corresponding number of loop iterations. To plot resistance circles, normalized resistances and labels are read in. Reactive circles use angular coordinates for each z = 0 + jx point, and labels.

SUB Splot(I,J)

The data-base is first transformed, if necessary, into S-parameters and then the appropriate parameter is plotted in rectangular coordinates from row 1 to row Count of the data-base. LINE TYPE 9 is used to put tic marks at each plotted point but GRAPHICS is left in LINE TYPE 1 (solid line) when the plot is complete. Note that if Smith was not called before Splot, a SHOW statement is necessary in Cktanalysis to scale the plotting area to the desired dimensions, e.g., SHOW -1,1,-1,1 for a unit-radius plotting area.

SUB Gammaz(Option,U,V,R,X)

The subroutine uses the following standard formulas:

$$\Gamma = \frac{(R^2 + X^2 - Z_o^2) + j(2XZ_o)}{(R + Z_o)^2 + X^2}$$

$$Z = \frac{(1 - \Gamma_r^2 - \Gamma_i^2) + j(2\Gamma_i)}{(1 - \Gamma_r)^2 + \Gamma_i^2}$$

where $\Gamma = \Gamma_r + j\Gamma_i$ and $Z = R + jX$.

If either denominator is zero, the resulting Γ or Z is taken as $10^{10} + j0$.

SUB Zio(X(*),Rs,Xs,Rl,Xl,Rin,Xin,Rout,Xout)

The ABCD-parameters are obtained for X(*), and the input and output impedan are calculated by:

$$Z_{in} = \frac{AZ_L + B}{CZ_L + D} \qquad Z_{out} = \frac{DZ_S + B}{CZ_S + A}$$

If either denominator is zero, the resulting impedance is taken to be $10^{10} + j0$.

SUB Optimize(N,X(*))

The Davidon algorithm used by FARANT requires the initial values of the variables, a method of obtaining the value of the objective function, and the gradient vector. From successive steps in approximately the gradient direction, it updates a Jacobian matrix of second derivatives for a quadratic approximation to the objective function. At the start this matrix is set equal to an identity matrix by the subroutine, but after several steps begins to reflect more accurate second derivative information. The subroutine also chooses the value of the algorithm's quitting parameter to be 10^{-12}. The variable Eps, set equal to this value, is used by the algorithm as a measure of the machine accuracy. It was found experimentally that best performance was achieved for Eps in the range 10^{-10} to 10^{-15} depending somewhat on the particular optimization problem.

The taking of numerical derivatives requires a choice of a derivative stepsize Δx. This choice is at best a compromise in that too small a Δx loses too many digits in the evaluation of $f(x + \Delta x) - f(x)$, but too large a Δx produces a derivative which is itself inaccurate due to curvature of the function. Empirical studies again proved helpful in deciding on this parameter. It was found that a 10^{-5} fractional change in each variable produced excellent results for a variety of problems whose objective function was accurate to 10 or 11 digits It was also decided that this stepsize should not change as variables grew large or small during the optimization process so 10^{-5} of the <u>initial</u> value of each variable is used to evaluate the derivatives at every step--thus, the requirement that no variable is initially zero.

Davidon's algorithm is claimed to be optimally conditioned in the method by which it updates the second derivative matrix. It does not make line searches,

maintains quadratic termination, and strongly non-linear functions can be optimized with it. However, it happens on occasion that an optimization step produces a larger function value instead of a smaller one. In this case, the algorithm cuts that step in half enough times so as to produce a smaller value for the function, if possible. The subroutine does not print the values of the variables as this is happening so these steps are missing from the printout. Usually about 80 to 90% of the steps will, however, improve the objective functio until the optimization process has found a minimum.

SUB Nread(X(*)) SUB Pread(X(*))

Each of these subroutines searches its DATA statements for the frequency of analysis and installs the appropriate parameters in program storage units into X(*). Note that SUB Pread first zeroes the entire matrix whereas Nread only overwrites the noise representation.

If it is desired to label the data with some quantity other than frequency, e.g., bias or device type, then another pass-parameter could be added to the subroutine parameter list to facilitate identification of the proper set of data on each call. The search that looks for frequency should then be modified to look instead for the label passed to the subroutine by this additional pass-parameter.

REFERENCES

[1] S. Weinreb, "Course Notes for EE217-Microwave Networks," given Spring 1978, Univ. of California, Berkeley.

[2] W. C. Davidon, "Optimally Conditioned Optimization Algorithms Without Line Searches," Math. Programming, Vol. 9 (1975), pp. 1-30.

[3] Larry D'Addario, "VAX Terminal Simulator for Micromodem," NRAO Program for Apple II Computer, Nov. 23, 1980, on NRAO Library Disk 1.1.

[4] D. C. Hayes Associates, Inc., "Micromodem II Owner's Manual," 3rd ed., 1979.

[5] John Granlund, private communication, Nov. 20, 1980.

[6] "S-Parameter Design," Hewlett-Packard Application Note 154, April 1972.

[7] R. W. Anderson, "S-Parameter Techniques for Faster, More Accurate Network Design," Hewlett-Packard Application Note 95-1, Feb. 1967.

[8] H. A. Haus, "Representation of Noise in Linear Two-Ports," Proc. IRE, Vol. 48, pp. 69-74, Jan. 1960.

[9] H. Rothe and W. Dahlke, "Theory of Noisy Fourpoles," Proc. IRE, Vol. 44, pp. 811-818, June 1956.

[10] John Granlund, "Correction of Noise Parameter Measurements," NRAO Electroni Division Internal Memo, 30 Oct. 1978.

[11] R. P. Meys, "A Wave Approach to the Noise Properties of Linear Microwave Devices," IEEE Trans. Microwave Theory Tech., Vol. MTT-26, No. 1, pp. 34-37, Jan. 1978

[12] P. Penfield, "Wave Representation of Amplifier Noise," IRE Trans. Circuit Theory, Vol. CT-9, pp. 84-85, March 1962.

[13] R. Q. Twiss, "Nyquist's and Thevenin's Theorems Generalized for Non-Reciprocal Linear Networks," J. Appl. Phys., Vol. 26, No. 5, pp. 599-602, May 1955.

[14] John Granlund, "Lossy Transmission Lines," NRAO Electronics Division Memo, 31 July 1979.

[15] J. M. Rollett, "Stability and Power-Gain Invariants of Linear Two-Ports IRE Trans. Circuit Theory, Vol. CT-9, No. 1, pp. 29-32, March 1962