NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VIRGINIA

ELECTRONICS DIVISION INTERNAL REPORT No. 236

# FORTRAN VERSIONS OF FARANT

CHIU TAI LAW
(SUMMER STUDENT)

SEPTEMBER 1983

NUMBER OF COPIES: 150

# FORTRAN VERSIONS OF FARANT

Chiu Tai Law

# FORTRAN VERSIONS OF FARANT


## ABSTRACT

Two new versions of the program **FARANT** were written in **VAX FORTRAN.** They offer the flexibility of incorporating user's own **FORTRAN** program for specific problem; the power of **FARANT** in frequency analysis of two-port networks, computing the two-port and noise parameters in various representation; and optimization. In addition, the programs written in **VAX FORTRAN** allow data to be entered interactively, provide a simple plotting subroutine and job process information at the end of each run. The flow of programs can fully controlled by issuing control/c which halts the program and creates a data file storing current values of objective variables for optimization. In average, the **VAX FORTRAN** versions of **FARANT** run about sixty times faster than the **BASIC** version. This report is a guide for user who has experience in using the **BASIC** version of **FARANT** and is a supplement to the EDIR No. 217.

## 1.0 INTRODUCTION

**FARANT** is a program with many useful subroutines for analysing steady state ac microwave circuits; it offers optimization. It can be combined with user's program in solving problems. However, the **HP 9845** version of **FARANT** runs too slowly for extensive optimization problems. The two **FORTRAN** versions of **FARANT** were developed for this reason. These **FORTRAN** versions will be called **FARANT 1.0** and **FARANT 2.0**. They run about sixty times faster than the **HP 9845** version. Generally, **FARANT 1.0** runs faster than **FARANT 2.0**. Both allow the user to enter data interactively or through a data file. **FORTRAN** versions of **FARANT** also allow the user to store results in a data file or print them on the terminal.

The purpose of this report is to describe **FARANT 1.0** and **FARANT 2.0**. As in the **HP 9845** version, any user's statements written in **FORTRAN** can be used in the subroutine **CKTANALYSIS** to control the calculations. The user can enter his own library containing subroutines to be used with **FARANT**. All these and their differences from the **BASIC** version in storing two-port descriptions and passing parameters and entries of the subroutines will be discussed in the next section. Moreover, the **FORTRAN** versions are compared to the **BASIC** version in terms of complex number manipulation and programming. Although some special features of **VAX FORTRAN** will be described briefly in following sections, readers should be familiar with **FORTRAN IV.**

## 2.0 FARANT USER'S GUIDE

This portion of the report is intended to help the user to run **FARANT** on the **VAX.** Therefore, short descriptions with examples are given for those routines having great differences in **FORTRAN** and **BASIC** versions as readers are expected to be familiar with the **BASIC** version. The arrangement of the two-port and noise parameters, the location of **FARANT** in the **VAX** and the way **FARANT** is run are described. However, the detailed programming technique for using **VAX FORTRAN** and a survey of **FARANT** are beyond the scope of this section. Finally, the **FORTRAN** versions of the optimization program given in EDIR No. 217 are used for demonstrating some of the differences between the **BASIC** and **FORTRAN** version.

### 2.1 Conventions and Definitions

The units and two-port descriptions of the **FORTRAN** version are the same as the **BASIC** version. Hence, they are not repeated here. In the following sections, two-port identifier is refered to the matrix containing noise and two-port parameters for **FARANT 2.0** and to the matrix containing either noise parameters or two-port parameters for **FARANT 1.0**.

**FARANT** is used through the CALL statement.

CALL name of subroutine (argument list ...)

The called subroutine can be any one of the **FARANT** subroutines listed in the next subsection. In **FARANT 2.0**, the first argument is often the two- port identifier. In **FARANT 1.0**, the first and second arguments are the two -port and noise parameters, respectively. Therefore, two two-port arrays are needed in **FARANT 1.0**. The rest of the arguments can be constants or variables that are expressions of real, integer or complex data type and they can also be character strings. However, they must have the same data type and order as the argument list in the SUBROUTINE statement. Furthermore, some of the arguments are intended to be inputs, outputs or both. For those arguments used as output parameters, constants or expressions must not be assigned in the CALL statement, otherwise the program will be halted.

In **FARANT 1.0** the two-port parameters are stored in the first four elements of a complex (5x1) matrix and PSET is stored in the last element. The noise parameters are stored in the first four elements of a real (5x1) matrix and NSET is stored in the last element. In the subroutine CKTANALYSIS, two-port identifiers A through H are assigned to complex two-port parameters and A1 through H1 are assigned to noise parameters. Two-port parameters inside the two-port identifier are arranged as follows:

element(1) = two-port parameter (1,1)

element(2) = two-port parameter (1,2)

element(3) = two-port parameter (2,1)

element(4) = two-port parameter (2,2)

However, the noise parameters are stored in the same order as is used in the **BASIC** version.

For **FARANT 2.0**, noise and two-port parameters are stored in a real (4x4) matrix. The first two rows are loaded will two-port parameters. The first and second column of the third row contain the labels for two- port and noise parameters, respectively. Finally, the last row consists of noise parameters which are arranged in the same order as the **BASIC** version. The order of two-port parameters in the (4x4) matrix are as follows:

two-port parameter (1,1) = (element(1,1),element(1,2))

two-port parameter (1,2) = (element(1,3),element(1,4))

two-port parameter (2,1) = (element(2,1),element(2,2))

two-port parameter (2,2) = (element(2,3),element(2,4))

In other words, all the elements in the first and third columns of the first and second rows are the real parts of the two-port parameter. The rest of the elements in the first two rows are imaginary parts of the two-port parameters.

## 2.2 Short Descriptions of Subroutines in FARANT

In the descriptions below, the data type and type of argument are shown immediately following the argument. The items that apply only to **FARANT 1.0** are enclosed by braces,"{}", while those items applying only to **FARANT 2.0** are enclosed by brackets,"[]". Since similar descriptions can be found in the listing of **FARANT 1.0** and **FARANT 2.0**; and also in [1], only a few words will be used to describe the arguments in most of the subroutines. However, longer descriptions and examples are given for those subroutines that have been modified. All the complex and real variables are in double precision if they are not specified.

### 2.2.1 **Two-port Elements** -

#### 2.2.1.1 RLC Network -

Subroutine RLC creates an ABCD matrix for parallel or series RLC circuits placed in series or parallel.

Form: {**CALL RLC(X,Y,TYPE,R,L,C,PLACE,TAMB)**}

[**CALL RLC(Z,TYPE,R,L,C,PLACE,TAMB)**]

Arguments:

{X} -- Complex (5X1) for output, stores two- port parameters.

{Y} -- Real (5X1) for output, stores noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

TYPE -- One string for input, specifies series or parallel RLC network by accepting 'S' or 'P'.

PLACE -- One string for input, specifies whether network is to be placed in series or parallel by accepting 'S' or 'P'.

R,L,C -- 3 real numbers for input, specify values of resistance, inductance and capacitance.

TAMB -- 1 real number for input, specifies ambient temperature in degree Kelvin.

2.2.1.2 Lossless Transmission Line (TRLINE) -

Subroutine TRLINE computes ABCD parameters for any lossless line.

Form: **{CALL TRLINE(X,Y,ZG,LENGTH,K)}**

**[CALL TRLINE(Z,ZG,LENGTH,K)]**

Arguments:

{X} -- Complex (5X1) for output, stores two-port parameters.

{Y} -- Real (5X1) for output, stores noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

LENGTH -- 1 real number inputs the length of transmission line in inches.

K -- 1 real number inputs the product of the relative dielectric constant and the relative permeablility.

### 2.2.1.3 Lossy Transmission Lines (LOSSYLINE) -

Subroutine LOSSYLINE computes the impedance matrix with noise parameter for a lossy line.

Form: {**CALL LOSSYLINE(X,Y,ZGO,LENGTH,K,CATTN,DATTN,FO,TAMB)**}

[**CALL LOSSYLINE(Z,ZGO,LENGTH,K,CATT,DATT,FO,TAMB)**]

Arguments:

The first 4 {5} arguments are the same as for the subroutine **TRLINE.**

CATTN -- Real (input), is the attenuation in dB/in due to conductor losses.

DATTN -- Real (input), is the attenuation in dB/in due to dielectric losses.

FO -- 1 real number inputs the frequency at which CATTN and DATTN are measured.

TAMB -- 1 real number for input, is the ambient temperature.

### 2.2.1.4 Ideal Transformer (TF) -

Subroutine TF finds the ABCD parameters for an ideal transformer.

Form: {**CALL TF(X,Y,TURN1,TURN2)**}

[**CALL TF(Z,TURN1,TURN2)**]

Arguments:

{X} -- Complex (5X1) for output, stores the two-port parameters.

{Y} -- Real (5X1) for output, stores the noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

TURN1,TURN2 -- 2 real numbers for output, are the numbers of primary and secondary turns; only their ratio is significant.

#### 2.2.1.5 Controlled Sources (SOURCE) -

Subroutine SOURCE creates the impedance parameters for a voltage- or current- controlled voltage or current source.

Form: **{CALL SOURCE(X,Y,CONTROL,STYPE,GAIN,R1,R2,DELAY)}**

**[CALL SOURCE(Z,CONTROL,STYPE,GAIN,R1,R2,DELAY)]**

Arguments:

{X} -- Complex (5X1) for output, contains two-port parameters.

{Y} -- Real (5X1) for output, contains noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

CONTROL -- String for input, specifies a voltage- or current-controlled source by accepting 'V' or 'C'.

STYPE -- 1 character for input, specifies voltage or current source by accepting 'V' or 'C'.

R1 -- 1 real number inputs the resistance in port 1.

R2 -- 1 real number inputs the resistance in port 2.

GAIN -- 1 real number inputs the gain of the source.

DELAY -- 1 real number inputs the delay of the source in responding to the control in psec.

### 2.2.1.6 Measured Two-port Parameters (PREAD) -

The user enters two-parameters using the keyboard, and these are stored by subroutine PREAD. The two-port and noise parameters are first initialized at zero. It is therefore advisable to use PREAD before NREAD, which will be described next.

Form: {**CALL PREAD(X,Y)**}

[**CALL PREAD(Z)**]

Argument:

{X} -- Complex (5X1) for output, contains two-port parameters.

{Y} -- Real (5X1) for output, contains noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

Fig. 1 demonstrates the use of PREAD. <CR> means hitting the carriage return. The user's version of CKTANALYSIS passes the frequency at which data are to be entered to PREAD and PREAD prompts the user for the form of complex number representation and the two-port parameter type he will enter. Appropriate responses are "MPH" -- magnitude and phase or "RI" -- real and imaginary parts for the first question and PSET -- 1 through 5 -- for the second. At the first call of PREAD, it asks the user if the present PSET and form of complex number representation will be retained at each frequency. The user should type "Y" if he want to have the same PSET and form of complex noise number representation every time and the above two questions will not be asked again. Next, for the current frequency of data entry, the first two two-port parameters, parameter (1,1) and parameter (1,2), are requested. The user can type in four numbers, delimited by a comma or a space, e.g. xx,xx,xx,xx. If the data are whole numbers, it is not necessary to put a decimal point after each. The last two parameters are entered in the same way. After all the data are typed in, they will be re-printed and the user can edit his data by using the following symbols:

| Symbol | Action of the computer |
|---|---|
| 1 | Change real part of parameter (1,1) |
| 2 | Change imaginary part of parameter (1,1) |
| 3 | Change real part of parameter (1,2) |
| 4 | Change imaginary part of parameter (1,2) |
| 5 | Change real part of parameter (2,1) |
| 6 | Change imaginary part of parameter (2,1) |
| 7 | Change real part of parameter (2,2) |
| 8 | Change imaginary part of parameter (2,2) |
| P | Change PSET |
| A | Change all of the data |
| T | Change the form of complex number |
| Y | Store all the data |

If PREAD is re-called, it will give a prompt asking the user if he wants to keep the previous data. If the user type "Y", the previous data will be used and printed out on the terminal. However, those printed data may be different because they have been changed to the form and unit used in the program.

2.2.1.7 Measured Noise Parameters (NREAD And NLOAD) -

Subroutine NREAD accepts data interactively, while subroutine NLOAD receives data passed by the arguments. Neither initializes the two-port for noise parameters.

Form: {**CALL NREAD(Y)**}

[**CALL NLOAD(Z)**]

Arguments:

{Y} -- Real (5X1) for output, stores the noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

Fig. 2 demonstrates the use of subroutine NREAD. Data entry is similar to that for PREAD. First, the frequency at which the data are entered is printed. A prompt will ask for NSET, user's answers whehter the same NSET to be used for each frequency and then the four noise parameters. The user can enter data as was described previously and edit the data using the following symbols:

Fig. 1 Demonstration of PREAD.



```
 PLEASE ENTER DATA FOR FREQ. =   1.000000     GHz.


 TYPE "MPH" TO ENTER DATA IN POLAR FORM
 TYPE "RI" TO ENTER DATA IN RECTANGULAR FORM
> MPH                                                   <CR>

 WHAT IS PSET ? 2
 TYPE "Y" TO KEEP PSET AND FORM OF COMPLEX # THE SAME FOR THE RUN, ELSE TYPE "N'
> Y
                                                          < CR >
 PLEASE ENTER THE TWO PORT PARAMETERS AS FOLLOWS:

PHASE MUST BE IN DEGREE FOR POLAR FORM
                X(1,1)                                      X(1,2)
      MAG                   PH                 MAG                  PH
1,80,1,80                                                                  <CR>
                X(2,1)                                      X(2,2)
      MAG,                  PH,                MAG,                 PH
1,80,1,80
                                                                        CR
 AT    1.000000      GHz,
 X(1,1) = (  1.00000     ,  80.0000     )   X(1,2) = (  1.00000     ,  80.0000     )
 X(2,1) = (  1.00000     ,  80.0000     )   X(2,2) = (  1.00000     ,  80.0000     )
     PSET = 2.

 TYPE "Y" IF DATA ARE CORRECT
 TYPE "P" TO CHANGE PSET
 TYPE "1" TO CHANGE REAL PART OF X(1,1)
 TYPE "2" TO CHANGE IMAGINARY PART OF X(1,1)
 TYPE "3" TO CHANGE REAL PART OF X(1,2)
 TYPE "4" TO CHANGE IMAGINARY PART OF X(1,2) AND SO ON
 TYPE "A" TO CHANGE ALL OF THE DATA
 TYPE "T" TO CHANGE THE DATA TYPE (MPH OR RI)
> 1
                                                                      < CR>
 X(1) = ? 2

 AT    1.000000      GHz,
 X(1,1) = (  2.00000     ,  80.0000     )   X(1,2) = (  1.00000     ,  80.0000     )
 X(2,1) = (  1.00000     ,  80.0000     )   X(2,2) = (  1.00000     ,  80.0000     )
     PSET = 2.

 TYPE "Y" IF DATA ARE CORRECT
 TYPE "P" TO CHANGE PSET
 TYPE "1" TO CHANGE REAL PART OF X(1,1)
 TYPE "2" TO CHANGE IMAGINARY PART OF X(1,1)
 TYPE "3" TO CHANGE REAL PART OF X(1,2)
 TYPE "4" TO CHANGE IMAGINARY PART OF X(1,2) AND SO ON
 TYPE "A" TO CHANGE ALL OF THE DATA
 TYPE "T" TO CHANGE THE DATA TYPE (MPH OR RI)
> Y                                                                  < CR >
 0.3472963553338607       0.1736481776669303       2.000000000000000
 0.0000000000000000E+00    1.969615506024416      0.9848077530122081
 0.0000000000000000E+00   0.0000000000000000E+00   0.1736481776669303
 0.1736481776669303       0.0000000000000000E+00   0.0000000000000000E+00
 0.9848077530122081       0.9848077530122081       0.0000000000000000E+00
 0.0000000000000000E+00

 PLEASE ENTER DATA FOR FREQ. =   2.000000     GHz.
```

| Symbol | Action of the computer |
|---|---|
| 1 | Change noise parameter 1 |
| 2 | Change noise parameter 2 |
| 3 | Change noise parameter 3 |
| 4 | Change noise parameter 4 |
| 5 | Change NSET |
| A | Change all the data |
| Y | Store data |

Form: {**CALL NLOAD(X,Y,NSET,N1,N2,N3,N4)**}

[**CALL NLOAD(Z,NSET,N1,N2,N3,N4)**]

Arguments:

{X} -- Complex (5X1) for output, stores two-port parameters.

{Y} -- Real (5X1) for output, stores noise parameters.

[Z] -- Real (4X4) for output, is the two-port identifier.

NSET -- 1 integer for input, is the label of noise parameters.

N1,N2,N3,N4 -- 4 real numbers input noise parameters to be entered.

### 2.2.2 Interchanging Ports and Creating Branch Elements -

```
 PLEASE ENTER DATA FOR FREQ. =   1.000000     GHz.


 WHAT IS NSET (1 TO 8)? 1
 TYPE "Y" TO KEEP NSET THE SAME FOR THE RUN, ELSE TYPE "N"
> Y                                                                <CR>

 PLEASE ENTER THE NOISE PARAMETERS ON THE SAME LINE AS FOLLOWS:
     N(1),                 N(2),                 N(3),                 N(4)
1,2,3,4                                                                         <CR>
 AT    1.000000      GHz,
 N(1) =    1.0000      N(2) =    2.0000      N(3) =    3.0000      N(4) =    4.0000
     NSET = 1.

 TYPE "Y" IF DATA ARE CORRECT
 TYPE "A" TO CHANGE ALL OF THE DATA
 TYPE "1" TO CHANGE N(1)
 TYPE "2" TO CHANGE N(2)
 TYPE "3" TO CHANGE N(3)
 TYPE "4" TO CHANGE N(4)
 TYPE "5" TO CHANGE NSET
> 1
                                                         <CR>
 N(1) = ? 2
 AT    1.000000      GHz,
 N(1) =    2.0000      N(2) =    2.0000      N(3) =    3.0000      N(4) =    4.0000
     NSET = 1.

 TYPE "Y" IF DATA ARE CORRECT
 TYPE "A" TO CHANGE ALL OF THE DATA
 TYPE "1" TO CHANGE N(1)
 TYPE "2" TO CHANGE N(2)
 TYPE "3" TO CHANGE N(3)
 TYPE "4" TO CHANGE N(4)
 TYPE "5" TO CHANGE NSET
> Y                                                      <CR>
 0.0000000000000000E+00   0.0000000000000000E+00   0.0000000000000000E+00
  2.0000000000000000      0.0000000000000000E+00   0.0000000000000000E+00
  1.0000000000000000      2.0000000000000000E-03   0.0000000000000000E+00
 0.0000000000000000E+00   0.0000000000000000E+00   0.1892744706434237
 0.0000000000000000E+00   0.0000000000000000E+00   0.0000000000000000E+00
 1.3235360314390173E-02

 PLEASE ENTER DATA FOR FREQ. =   2.000000     GHz.


 TYPE "Y" TO KEEP THE OLD DATA.
 TYPE "N" TO ENTER NEW DATA.
> N
                                                         <CR>
 PLEASE ENTER THE NOISE PARAMETERS ON THE SAME LINE AS FOLLOWS:
     N(1),                 N(2),                 N(3),                 N(4)
2,3,4,5                                                                     <CR>
 AT    2.000000      GHz,
 N(1) =    2.0000      N(2) =    3.0000      N(3) =    4.0000      N(4) =    5.0000
     NSET = 1.

 TYPE "Y" IF DATA ARE CORRECT
 TYPE "A" TO CHANGE ALL OF THE DATA
 TYPE "1" TO CHANGE N(1)
 TYPE "2" TO CHANGE N(2)
 TYPE "3" TO CHANGE N(3)
```

Subroutine FLIP interchanges port 1 and port 2.

Form: {**CALL FLIP(X,Y)**}

[**CALL FLIP(Z)**}

Arguments:

{X} -- Complex (5X1) for input and output, stores the two-port parameters.

{Y} -- Real (5X1)for input and output, stores the noise parameters.

[Z] -- Real (4X4) for input and output, is the two-port identifier.

Subroutine BRANCH creates the description of a two-port containing port 1 of the two-port that was input either in series or in parallel.

Form: {**CALL BRANCH(X,Y,TYPE)**}

[**CALL BRANCH(Z,TYPE)**]

Arguments:

{X} -- Complex (5X1) for output and input, contains the two-port parameters.

{Y} -- Real (5X1) for output and input, contains the noise parameters.

[Z] -- Real (4X4) for output and input, is the two-port identifier.

TYPE -- String for input, specifies parallel or series branching by accepting 'P' or 'S'.

### 2.2.3 **Cascading, Paralleling and Putting Two-port in Series -**

In cascading two networks, subroutine CAS uses their ABCD parameters and noise parameters 1; PSET = NSET = 1. Subroutine SER puts two networks in series, using their admittance matrices and noise parameters 2, and leaving PSET = NSET = 2. Subroutine PAR puts two networks in parallel, using their impedance matrices and noise parameters 3, and leaving PSET = NSET= 3.

Form: {**CALL CAS(X,Y,A,A1)**

**CALL SER(X,Y,A,A1)**

**CALL PAR(X,Y,A,A1)**}

[**CALL CAS(Z,B)**

**CALL SER(Z,A)**

**CALL PAR(Z,A)**]

Arguments:

{X} -- Complex (5X1) for input and output, carries the two-port parameters for one of the networks and returns the resulting two-port parameter.

{Y} -- Real (5X1) for input and output, carries noise parameters for X-network and returns the resulting noise parameters.

{A} -- Complex (5X1) for input, contains two-port parameters for the other network.

{A1} -- Real (5X1) for input, contains noise parmeters for the other network.

[X] -- Real (4X4) for input and output, is the two-port identifier for one of the networks.

[A], -- Real (4X4) for input, is the two-port identifer for
[B] the other network.

### 2.2.4 Transforming Two-port and Noise Parameters -

Subroutine MTRANS performs transformation between two-port parameters. Subroutine NTRANS performs transformation between noise parameters and may also call for two-port parameter transformations.

Form: {**CALL MTRANS(X,N)**
**CALL NTRANS(X,Y,NSET)**}
[**CALL MTRANS(Z,N,KFLAG)**
**CALL NTRANS(Z,NSET,KFLAG)**]

Argument:

{X} -- Complex (5X1) for input and output, stores the two-port parameters to be transformed and is loaded with transformed two-port parameters.

{Y} -- Real (5X1) for input and output, stores the noise parameters to be transformed.

[Z] -- Real (4X4) for input and output, is the two-port identifier and is load with transformed parameters in return.

KFLAG -- 1 integer for input, is used for identifying whether Z or X will be used as pass parameter. If it is zero Z will be the argument. If KFLAG is one, X will be used as pass parameter and another entry should be used by issuing "CALL MTRANS(X,N,1)" for MTRANS and "CALL NTRANS1(X,Y,NSET,1)" for NTRANS.

PSET -- 1 integer for input, is the label of the desired two-port parameters. If these two-port parameters are undefined, "IFLAG" ("NOGO" in the **BASIC** version) is set to one.

NSET -- 1 integer for input, is the label of the desired noise parameters. If these noise parameters are undefined, the two-port description may be changed to the other form.

### 2.2.5 Saving Circuit Parameters -

Subroutine SAVECKT stores all the two-port descriptions at each frequency providing that all parameters are stored in the same type.

Form: {**CALL SAVECKT(X,Y,N,NSET,KFACT)**}

[**CALL SAVECKT(Z,PSET,NSET,KFACT)**]

Augrments:

{X} -- Complex (5X1) for input, stores the two-port parameters.

{Y} -- Real (5X1) for input, stores the noise parameters.

[Z] -- Real (4X4) for input, is the two-port description.

N -- 1 integer for input, is the desired type of two-port parameters to be stored in data base, DB. It can have a value from -5 to 5 and is the same

PSET in the **BASIC** version.

NSET -- 1 integer for input, names the desired type of noise parameters to be stored. It can have a value from -8 to 8.

KFACT -- 1 real number for output and input, is the stability factor. If it is less than zero, no KFACT will be computed and data base will receive a zero k-factor. In order to receive a value, KFACT must be a non-negative valued variable.

### 2.2.6 Noise Temperature and Gain Analysis -

Subroutine NPERFORM (equivalent to SUB NPERFORMANCE of the **BASIC** version) is the subroutine in which the noise temperature and gain of a two-port network are computed. Its results will be stored in data base as well as output as pass parameters. Therefore, the type of gain requested should be the same as for each call. NPERFORM should be called after SAVECKT and use the same two-port identifier as for the SAVECKT.

Form: {**CALL NPERFORM(X,Y,GTYPE,ZS,ZL,GAIN,TN)**}

[**CALL NPERFORM(Z,GTYPE,ZS,ZL,GAIN,TN)**]

Arguments:

{X} -- Complex (5X1) for input, contains two-port parameters and should be the same parameters which are used in SAVECKT.

{Y} -- Real (5X1) for input, contains noise parameters.

[Z] -- Real (4X4) for input, is the two-port identifier for the two-port to be analysed.

GTYPE -- 1 integer for input; specifies transducer, power, available or maximum available gain by accepting 1, 2, 3 or 4 respectively. If it is zero, gain will not be calculated.

ZS -- 1 complex number for input, is the source impedance and is entered as (RS,XS).

ZL -- 1 complex number for input, is the load impedance driven by the two-port and should be assigned in the form of (RL,XL).

GAIN -- 1 real number for output and input, receives the gain in dB for GTYPE equal to one to four.

TN -- 1 real number for output and input, will receive the noise temperature if RS is positive, the two-port has noise parameters and TN is assigned a non-negative value.

### 2.2.7 Reflection and Impedance Calculations -

Subroutine GAMMAZ performs the conversion between reflection coefficient and impedance.

Form: **CALL GAMMAZ(OPT,U,V,R,X)**

Arguments:

OPT -- 1 integer for input, is used for indicating the required type of conversion:

-2 impedance to reflection coefficient (rectangular form)

-1 impedance to reflection coefficient (polar form)

0 nothing will be done

1 reflection coefficient (polar form) to impedance

2 reflection coefficient (rectangular form) to impedance

R,X -- 2 real numbers input or output, are the real and imaginary parts of the impedance in the form of R+jX.

U,V -- 2 real numbers input or output, are the reflective coefficient either in rectangular form as U+jV or in polar form as U $\angle \underline{V}$.

ZIO computes the input and output impedances for a two-port network.

Form: {**CALL ZIO(X,Y,ZS,ZL,ZIN,ZOUT)**}

[**CALL ZIO(Z,ZS,ZL,ZIN,ZOUT,KFLAG)**]

Arguments:

{X} -- Complex (5X1) for input, stores the two-port parameters.

{Y} -- Real (5X1) for input, is the noise parameters.

[Z] -- Real (4X4) for input, is the two-port identifier.

ZS,ZL -- 2 complex numbers input source and load impedances in the form of (RS,XS) and (RL,XL) respectively.

ZIN, -- 2 complex numbers input the input and output
ZOUT impedances in the form of (RIN,XIN) and (ROUT, XOUT).

### 2.2.8 **Printing Circuit Parameters** -

Subroutine PRT can print five forms of two-port parameters and eight forms of noise parameters. The format of the printout is the same as for the **BASIC** version. After the printing is finished, the terminal screen will be frozen until the carriage return is hit.

Form: **CALL PRT(PSET,NSET)**

Parameters:

PSET -- 1 integer for input, can have a value -5 to 5.
If |PSET| = 1 to 5, all the two-port parameters in data base will be transformed to type |PSET|.
If PSET > 0, two-port parameters will be printed.
If PSET < 0, two-port parameters are not printed.
If PSET = 0, nothing will be done.

NSET -- 1 integer for input, can have a value -8 to 8.
if |NSET| = 1 to 8, all the noise parameters in data base will be transformed to type |NSET|.
If NSET > 0, noise parameters will be printed.
If PSET < 0, noise parameters are not be printed.
If NSET = 0, nothing will be performed.

### 2.2.9 **Plotting in FARANT** -

Subroutine PLOT performs a simple point plotting. Its maximum capacity is to plot ten curves with 70 points in each curve on the same plot with or without x-axis and y-axis transposed. PLOT finds the appropriate scale to accomodate all the points on the plot or use the range specified by the user. It also stores the previous plots and puts them together with the present plot. At the end of each plot, the terminal screen will be frozen until the carriage return is hit.

Form: **CALL (X,Y,CHA,XMIN,XMAX,YMIN,YMAX,MODE,M,N,VAS,HAS,TITL)**

Arguments:

X -- Real (NX1) for input, has a maximum dimension of 70 and contains x-coordinate values, e.g. frequency.

Y -- Real (MXN) for input, has a maximum dimension of 10X70, and contains the plotting values for functions, e.g. S parameters.

XMIN,XMAX -- 2 real numbers input the minimum and maximum values of the x-axis. If they are equal,auto-scaling will be performed.

YMIN,YMAX -- 2 real numbers input the minimum and maximum values of the y-axis. If they are equal, auto-scaling will be performed.

MODE -- 1 integer for input, names the form of plotting and has a value -3 through 3. If it is negative, the plot will be saved without display. For non-negative valued MODE, the plot will be saved and displayed.

If |MODE| = 0, new plot will be made and the old image will be erased.

If |MODE| = 1, new plot will be plotted with old image using old scale providing that the previous calls of PLOT have MODE equal to 1 or 0.

If |MODE| = 2, plot will be transposed with old image erased.

If |MODE| = 3, functions for |MODE| = 1 and 2 are performed providing that the previous calls for PLOT have |MODE| equal to 2 or 3.

CHA -- M strings for input, contains the character used for plotting each curve. Its dimension, M, must agree with the number of curves.

M -- 1 integer inputs the number of curves to be plotted.

N -- 1 integer inputs the number of points in each curve.

VAS -- Strings for input, stores the vertical axis label locating at the left hand side of the plot.

HAS -- Strings for input, stores the horizontal axis label locating at the right hand side of the plot. For both VAS and HAS, maximum 21 characters can be used for the label.

TITL -- Strings for input, stores the title below the plot. It can carry 76 characters at most.

On the following pages, a demonsration of PLOT is shown. The labels and title are assigned by means of data statements. The first subscript of Y is used as the curve identifier. In this demonstration, two plots, one plot with sine and cosine funtions together; and the other with tangent function in transposed position, are made. The intersection of curves are indicated by "X".

```
C
      DIMENSION X(70),Y(2,70),A(50),B(50)
      CHARACTER CHA(2)/'*','1'/,VAS*21/'Y AXIS'/,HAS*21/'X AXIS'/
      CHARACTER TITL*76/' * - SIN FUNCTION 1 - COS FUNCTION'/,CH*1/'2'/
      CHARACTER TITL1*76/' * - SIN 1 - COS 2 - TAN'/
      DO 10 I = 2,140,2
      X(I/2) = FLOAT(I)/10.
      Y(2,I/2) = COS(X(I/2))
   10 Y(1,I/2) = SIN(X(I/2))
      READ(5,21) MODE1
      CALL PLOT(X,Y,CHA,1.,1.,1.,1.,MODE1,2,70,VAS,HAS,TITL)
      DO 20 I = 1,50
      A(I) = FLOAT(I)/10.
   20 B(I) = TAN(A(I))
      READ(5,21) MODE
   21 FORMAT(I2)
      CALL PLOT(A,B,CH,1.,1.,-10.,10.,MODE,1,50,VAS,HAS,TITL1)
      STOP
      END
```

```
$ RUN PLOT
Ø            (MODE)
Y 999571     ***                    1111     ***                     111     * X
  89968|1   *   *                  1    1   *   **                  1   1  **
A 79979| 1 *     *                1      1 *                       1     1     A
X 69989|  X       *                      *        *               1       X    X
I 60000| *                       1         1       *             1       *     I
S 50011|   1       *            1       *                                  1   S
  40021|*   1                               1       *           1       *
  30032|            *          1       *     1                              1
  20043*     1                                       *         1       *
  10053|             *        1       *       1                              1
     64|      1                                       *       1       *
  -9925|              *      1       *         1
 -19915|       1                                       *     1       *
 -29904|               *    1       *           1
 -39893|        1       *                               *   1       *
 -49883|                   1       *             1       *         *
 -59872|         1       *        *                        1
 -69861|          1       1                       1       X       *
 -79851|                 1*      *                 1     1 *     *
 -89840|           11   1  *    *                   1   1   *   *
 -99829+---------+---111---+****-----+---------+-----111-+---***---+---------+
E -5    200    2171      4143      6114      8086     10057     12029     14000
 * - SIN FUNCTION 1 - COS FUNCTION                                          E -3
 TYPE ANY CHARACTER TO CONTINUE. >
```

```
2            (MODE)
Y  1000|                                   22                                   X
   3450|                                    X
A  5900|                                     X2                                 H
X  8350|                                       X                                X
I 10800|                                        2 2 2
S 13250|                                                2      2                I
  15700|
  18150|        2           2    2
  20600|                           2 2
  23050|                              222
  25500|                                22
  27950|                                 2X
  30400|                                   X
  32850|                                  2X
  35300|                                    22
  37750|                                     22
  40200|                                      222
  42650|                                        2 2
  45100|                                              2    2              2
  47550|
  50000+---------+------2--+--2------+---------+---------+---------+---------+
E -3 -10000   -7143     -4286     -1429       1429      4286      7143     10000
 * - SIN 1 - COS 2 - TAN                                                      E -4
 TYPE ANY CHARACTER TO CONTINUE. >
FORTRAN STOP
```

2.2.10 **Optimization** - Subroutines OPTIMIZE, CKTANALYSIS and FARSTART involve in optimization. However, OPTIMIZE does the decision-making for the optimization. (For detail, see [2])

Form: **CALL OPTIMIZE(N,X)**

Arguments:

N -- 1 integer inputs the number of variables to be optimized to get a minimum objective function value. The setup in **FARANT** allows at most 24 variables to be used.

X -- Real (Nxl) for input and output, contains initial objective variable values as input and receives final values which produce minimum objective function value.

Subroutine CKTANALYSIS contains the user's defined objective function which has to be minimized.

Form: **CALL CKTANALYSIS(X,FVAL,OPT)**

Arguments:

X -- Real (24X1) input the objective variable values for evaluation of FVAL

FVAL -- 1 real number outputs the objective function value corresponding to each set of x entered.

OPT -- 1 integer for output, indicates whether other things should be done besides computing FVAL. The user can make use of OPT to control printing and plotting of initial or final objective values.

Subroutine FARSTART coordinates the operation between OPTIMIZE and CKTANALYSIS. Further, it gets the initial guesses and the user's reponse to decide whether optimization to be used interactively. Therefore, the user need not modify **FARANT** to run optimization. The trapping of control/c and fetching of job process information are also done in FARSTART.

Form: **CALL FARSTART**

Arguments: none

On the following page, a printout shows questions asked by FARSTART before optimization.

```
 TYPE "Y" TO HAVE OPTIMIZATION
 TYPE "N" TO DO NORMAL CIRCUIT ANALYSIS
> Y
 WHAT IS THE NUMBER (INTEGER) OF PARAMETERS TO BE OPTIMIZED ? 4
 PLEASE ENTER THE INITIAL GUESSES OF:
 CAUTION: USE NO ZEROS
 PARAMETERS #   1 = ? 15
 PARAMETERS #   2 = ? -2
 PARAMETERS #   3 = ? 3
 PARAMETERS #   4 = ? 5
 DATA ENTERED ARE AS FOLLOWS:
   15.00000        -2.000000        3.000000        5.000000
 TYPE "Y" IF DATA ARE CORRECT
 TYPE "N" TO CHANGE THE SET OF DATA
> Y
```

### 2.2.11 Job Process Information -

PROCESS_INFO fetches the information for cpu time, buffered I/O, direct I/O, and page faults and calls a system routine using FORTRAN language. (For detail, see [3] and [5])

Form: **CALL PROCESS_INFO(ABS_VALUES,INCR_VALUES)**

Arguments:

ABS_VALUES -- Integer (4X1) for output and input, gives the acculumative cpu time, I/O counts and page faults in one terminal session.

INCR_VALUES -- Integer (4X1) for output, gives the increment of cpu time, I/O counts and page faults for one job. In order to get them, PROCESS_INFO must be called twice.

### 2.2.12 **Control/c Trapping -**

Subroutines ENABLE_CTRLC and CTRLC_OUT are used for enabling control/c trapping and writing a file when control and c buttons are hit together, respectively. The control/c trapping is enabled by calling a system routine, QIOW. The output of the subroutine CNTRL_OUT is stored in a file FIRR.DAT which contains initial guesses and current values of objective variables and objection function. The purpose of these subroutines is to provide the user a way to stop the optimization without losing data. Every time the program halted by a control/c, "Abnormal Exit" will be printed on the terminal screen. If the user want to continue execution, they can type in the current values in FIRR.DAT as initial guesses and run the program again. For detailed descriptions and operation of the system routine, readers should see [3], [5] and [4].

Form: CALL ENABLE_CTRLC

CALL CTRLC_OUT

Arguments: none

### 2.2.13 **Lower Level Subroutines -**

Lower level subroutines are used by the subroutines in **FARANT.** Normally, the user will not use them.

#### 2.2.13.1 Matrix Handling Routines -

Matrix addition are done by ADD and ADD1. These routines are used by optimization and operate on real, linear matrix

Form: CALL ADD(X,C,Y,D,Z,N)

CALL ADD1(X,Y,Z,N)

Matrix multiplication of 2x2 complex matrices is performed by MUT.

Form: CALL MUT(X,Y)

Scalar product are performed by SCAL and COP which operate on real, linear matrix and 2X2 complex matrix respectively.

Form: CALL SCAL(X,C,N)

CALL COP(X,Y,C)

ADJ and DETT are written for MTRANS to find the inverse of a matrix. ADJ and DETT find the adjoint and determinant of a 2x2 complex, respectively.

Form: CALL ADJ(X)

DETT(Y)

Other routines are COPY, DOT and ADIN. COPY equates one real, linear matrix to the other. DOT computes the dot product of 2 real, vectors. In the subroutine ADIN, all the diagonal elements of a 2x2 complex matrix is added to one.

```
Form: CALL COPY(X,Y,N)
      DOT(X,Y,N)
      CALL ADIN(Y)
```

2.2.13.2 Routines Called by Optimization -

Subrouitnes GRAD and GRADIENT compute the gradient at a point of the objective function. However, GRAD takes three points to find a gradient. Hence, it is slower than GRADIENT which takes two points to compute a gradient. PVARS prints the intermediate steps of optimization.

```
Form: CALL GRAD(X,G,FVAL,N)
      CALL GRADIENT(X,G,FVAL,N)
      CALL PVARS(X,FVAL,N)
```

2.2.13.3 Routines Called by PLOT -

Auto-scaling of PLOT is done by FACTOR and SCALE. SCALE finds the scale which can include all the points. FACTOR computes the exponent for the label of the scale.

```
Form: CALL SCALE(Y,N,M,MAX,MIN)
      CALL FACTOR(MAXD,MIND,I)
```

#### 2.2.13.4 Miscellaneous Routines -

KCALC is called by SAVECKT to find the k-factor.

Form: CALL KCALC(Z,KDONE,KFACT)

REDIM combines the two-port and noise parameters into one matrix. REDIM1 separates one two-port matrix into two-port and noise parameters matrices. They are used in **FARANT 2.0** for those routines using complex number extensively.

Form: CALL REDIM(X,Y,Z)

CALL REDIM1(Z,X,Y)

Finally, POLAR converts rectangular coordinates into polar coordinates.

Form: CALL POLAR(X,Y)

### 2.3 Setup and Environment in VAX

**FARANT 1.0** and **2.0** are stored in different subdirectories of the **VAX.** In order to run them in the **VAX,** three commands -- FOR, LINK, and RUN -- have to be issued and two files will be created, besides the source file. The following paragraphs describe the above areas.

### 2.3.1 Pieces of FARANT in VAX -

**FARANT 1.0** AND **2.0** are stored under subdirectories [SW.FARANT1] and [SW.FARANT2] on device DBA0. The following file names are used in both subdirectories.

| Name.type | Description |
|---|---|
| FLIB.FOR | It is the entire **FORTRAN** code of **FARANT** without CKTANALYSIS and main program. |
| FLIB.OLB | It is the compiled version of FLIB.FOR stored in object module library. It is created by giving the command: LIBRARY/CREATE FLIB.OLB FLIB.OBJ or in short form as LIB/CRE FLIB FLIB. |
| FMAIN.FOR | It contains the **FORTRAN** code of the main program and initial setup of the subroutine CKTANALYSIS. |

A command file, RUNFRT.COM, is stored under the directory [SW]. Its aim is to help the running of **FARANT**.

Putting **FARANT** in the object module library enables the user to change the subroutines easily. The command, LIB/REPLACE, will replace some or all of the modules insides the library. Another command, LIB/INSERT, will insert more modules into the library. By this way, the user needs not compile the whole library again for mistakes in some of the library modules. For examples, to correct mistakes in MTRANS and NTRANS of the FLIB.OLB, the user only need to copy NTRANS and MTRANS into another file, e.g.

FILE. Then the user modifies these routines and types LIB/REPLACE FLIB FILE to replace modules. Other commands are LIB/LIST, LIB/DELETE and LIB/EXTRACT. In Fig. 3 and 4, the libraries of **FARANT 1.0** and **2.0** are listed using LIB/LIST command. A good summary of these commands can be found in [6]. Another advantage of putting **FARANT** in the library is the saving of storage space and time. Each time the **FARANT** library linked to user's program, the library will be searched for those unresolved subroutines referenced by the user's program. Therefore, only the routines called by him will be linked and copied. However, the user must put libraries in right order, if he have more than one libray. For example, a library, LIB1, containing modules A, B and C which calls modules D, and E in another library, LIB2. Then LIB1 must precede LIB2 in the LINK command -- LINK user's file + LIB1/LIB + LIB2/LIB.

### 2.3.2 **Running of FARANT in VAX** -

The user has to use the following command to copy **FARANT** to his working area:

```
For FARANT 1.0 COPY DBA0:[SW.FARANT1]FLIB.*;* FLIB.*;*
               COPY DBA0:[SW.FARANT1]FMAIN.*;* FMAIN.*;*
For FARANT 2.0 COPY DBA0:[SW.FARANT2]FLIB.*;* FLIB.*;*
               COPY DBA0:[SW.FARANT2]FMAIN.*;* FMAIN.*;*
```

Before using **FARANT** with user's programs, the user has to copy FMAIN.FOR to his programs. For instance, he wants to write his

**Fig. 3 List of modules in FARANT 1.0**

```
Directory of OBJECT library SYS$SYSDEVICE:[SW.FARANT1]FLIB.OLB;2 on 16-AUG-1983
14:18:06
Creation date:  25-JUL-1983 17:24:06       Creator:  VAX-11 Librarian V03-00
Revision date:  13-AUG-1983 14:12:48       Library format:   3.0
Number of modules:      43                 Max. key length:  31
Other entries:          44                 Preallocated index blocks:      49
Recoverable deleted blocks:      10        Total index blocks used:         6
Max. Number history records:      20       Library history records:        15

ADD
ADD1
ADIN
ADJ
BRANCH
CAS
COP
COPY
CTRLC_ROUT
DETT
DOT
ENABLE_CTRLC
FACTOR
FARSTART
FLIP
GAMMAZ
GRAD
GRADIENT
KCALC
LOSSYLINE
MTRANS
MUT
NLOAD
NPERFORM
NREAD
NTRANS
OPTIMIZE
PAR
PLOT
POLAR
PREAD
PROCESS_INFO
PRT
PVARS
RLC
SAVECKT
SCAL
SCALE
SER
SOURCE
TF
TRLINE
ZIO
$
```

Fig. 4 List of modules in FARANT 2.0

```
Directory of OBJECT library SYS$SYSDEVICE:[SW.FARANT2]FLIB.OLB;1 on 16-AUG-1983
14:20:03
Creation date:   1-AUG-1983 10:58:47      Creator:  VAX-11 Librarian V03-00
Revision date:  13-AUG-1983 13:58:21      Library format:   3.0
Number of modules:      45                Max. key length:  31
Other entries:          49                Preallocated index blocks:      49
Recoverable deleted blocks:       9       Total index blocks used:         6
Max. Number history records:       20     Library history records:         8

ADD
ADD1
ADIN
ADJ
BRANCH
CAS
COP
COPY
CTRLC_ROUT
DETT
DOT
ENABLE_CTRLC
FACTOR
FARSTART
FLIP
GAMMAZ
GRAD
GRADIENT
KCALC
LOSSYLINE
MTRANS
MUT
NLOAD
NPERFORM
NREAD
NTRANS
OPTIMIZE
PAR
PLOT
POLAR
PREAD
PROCESS_INFO
PRT
PVARS
RDM
RDM1
RLC
SAVECKT
SCAL
SCALE
SER
SOURCE
TF
TRLINE
ZIO
$
```

program in MYFILE.FOR. If he is in the EDT editor, he can use the command, INCLUDE FMAIN.FOR. When he is in the SOS editor, he can use the command C100 = FMAIN.FOR. In case the user does not copy **FARANT** into his work area; he must use the full name of the file, e.g. DBA0:[SW.FARANT1]FMAIN.FOR for **FARANT 1.0.**

To run **FARANT,** the user can execute RUNFRT.COM which contains all the commands needed to run **FARANT** by issuing the command:

@RUNFRT MYFILE

where MYFILE is user's program

This command carries out a procedure. Those users want to understand the command proccedure should consult [8]. All the versions of FIRR.DAT which are generated by CTRLC_OUT are deleted. If these files do not existed, computer will give a warning and continue execution. First, the user is asked if he wants to have program listings. If he responds by typing "YES" or even "Y", a listing of MYFILE will be generated; otherwise he can hit return and skip the question. Then the user is asked if he wants to generate all symbols needed for the debugger. He can answer the question in the way as the first question. Then MYFILE is compiled. Afterwards, a prompt asking if the user uses another library with **FARANT.** If he does, he types the name of his library; otherwise he skips the question by hitting the carriage return. The procedure continues and links MYFILE to FLIB and user's library, if any. Then, all the object files and old versions will be deleted. The procedure will request the user to indicate where the output will be sent and where the input will be read. The action of the procedure corresponding to user's reply is listed as follows:

| <u>User's reply</u> | <u>Action of the program</u> |
|---|---|
| Type in a file name for outputing results; type in a file name for inputing data | run the prgram **FARANT** by reading data from the input file and writing results to the output file |
| Type in a file name for an output file; answer the prompt about input file by hitting a cariage return | the result will be the same as asnwering the two questions by hitting the return (the user must type "RUN MYFILE" and enter data interactively to run the program) |
| Type in a file name for the input file; answer the question about the output file by hitting the carriage | run the program **FARANT** by reading data from the input file and writing to the terminal screen |

return

| | |
|---|---|
| Answer the two questions about input and output file by hitting the carriage return | exit the procedure (The user must run the program interactively by typing "RUN MYFILE" and data. The results will be printed on the terminal screen.) |

For all of the modes listed, the execution of the program can be halted by typing control and c buttons simultaneously. A listing of RUNFRT.COM is shown in Fig. 5. If the user has not copied **FARANT** into his directory, he modifies RUNFRT.COM by prefixing [SW.FARANT1] or [SW.FARANT2] to the FLIB/LIB in the link statement for **FARANT1** or **FARANT2** respectively, e.g. [SW.FARANT1]FLIB/LIB or [SW.FARANT2]FLIB/LIB.

```
$ DEL FIRR.DAT;*
$ INQUIRE LIS "ENTER 'YES' IF YOU WANT TO HAVE A PROGRAM LISTING"
$ INQUIRE DEB "ENTER 'YES' IF YOU WANT TO DEBUG YOUR PROGRAM"
$ IF LIS THEN SEL="/LIS"
$ IF DEB THEN SEL=SEL+"/DEB"
$ FOR'SEL' 'P1'
$ IF P2 .EQS.  "" THEN INQUIRE P2 " ENTER NAME OF LIB USED OTHER  THAN
FLIB;  IF YOU DON'T HAVE, HIT RETURN"
$ IF P2 .NES.  "" THEN P2=P2+"/LIB+"
$ IF DEB THEN SEP="/DEB"
```

```
$ LINK'SEP' 'P1'+'P2'FLIB/LIB

$ DEL 'P1'.OBJ;*

$ PUR 'P1'.*

$ WRITE SYS$OUTPUT "IF YOU WANT INTERACTIVE INPUT AND  OUTPUT,  ANSWER
FOLLOWING PROMPTS"

$ WRITE SYS$OUTPUT "BY HITTING RETURN  AND  TYPE  'RUN  program  name'
AFTER A '$' APPEARS."

$ INQUIRE FILE " ENTER NAME OF THE FILE IN  WHICH  THE  OUTPUT  TO  BE
STORED"

$ IF FILE .EQS.  "" THEN GOTO NEXT

$ ASS/USERMODE 'FILE' FOR006

$ NEXT:

$ INQUIRE FILE1 " ENTER NAME OF THE FILE IN WHICH DATA FOR UNIT  5  IS
STORED"

$ IF FILE1 .EQS.  "" THEN EXIT

$ ASS/USERMODE 'FILE1' FOR005

$ RUN 'P1'
```

Fig. 5 Listing of RUNFRT.COM.

The shortcoming of this procedure is that the user cannot enter data interactively inside a procedure; otherwise an end of file will be detected on logical unit 5. Therefore, the user must get out of RUNFRT.COM and type "RUN MYFILE" to run **FARANT** interactively. Normally, unit 5 and unit 6 are default to read data and write results on the terminal screen. However, RUNFRT.COM use the command ASSIGN/USER_MODE to change the default so that unit 5 and 6 are assigned to an input and output files temporarily until the execution of a program or procedure is completed.

## 2.4 Example Using Optimziation

On the following pages, two listings and outputs of **FORTRAN** programs which have the same function as the optimization program listed in [1] and use the **FORTRAN** versions of **FARANT** are shown in order to contrast some of the significant differences. The circuit in [1] is reproduced in Fig. 6e. The purposes of this program were to maximize gain and input return loss, minimize the noise temperature and make k-factor greater than one. The objective function was constructed as follows:

$$FVAL = 25/|S21|^2 + 10|S11|^2 + TN/50 - EXP(10X(1-K))$$

Four circuit elements were used as objective variables under the following contraints:

| **Variable** | **Constraint** | **Initial Values** | **Constraining function** | **Initial X(i)** |
|---|---|---|---|---|
| LIN | any value | 15 nH | LIN=X(1) | 15 |
| LFB | .2<LFB<2 | .466 nH | LFB=arctan(X(2))/100 +1.1 | -2 |
| ROUT | ROUT>10 | 30 Ohms | Rout=10+exp(X(3)) | 3 |
| LOUT | LOUT>0 | 25 nH | LOUT=X(4)*X(4) | 5 |

The programs listed on the following pages are both having the name CIR.FOR under subdirectories [SW.FARANT1] and [SW.FARANT2].

```
C**********************************************************************
C
C      THIS IS THE MAIN PROGRAM OF THE FARANT
C
C**********************************************************************
C
C  ASSIGN COMMON DATA BLOCK TO :
C  IFLAG -- INDICATES THE SUCCESS OF AN OPERATION BY HAVING A VALUE
C           ZERO (INTEGER)
C  ZO -- CHARACTERISTIC IMPEDENCE (REAL)
C  F -- FREQUENCY (REAL)
C  ICOU -- INDICATES THE SIZE OF DB (INTEGER)
C  DB -- DATA BASE FOR STORAGE OF DATA TO BE PRINTED OR PLOTTED (REAL)
C
      DOUBLE PRECISION ZO,DB,PI,F
      CHARACTER*8 HMS,DMY*9
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,F,ICOU,DB(101,18)/B1/PI
      PI = 3.14159265358979323 8D0
      CALL FARSTART
      CALL DATE(DMY)
      CALL TIME(HMS)
      WRITE(6,10) HMS,DMY
   10 FORMAT('  TIME: ',A8,49X,'DATE: ',A9)
      STOP 'SUCCESSFUL EXIT                         FARANT VERSION 1.0'
      END
C************************************************************************
C
C  FUNCTION: CKTANALYSIS
C
C  INPUT: X,OPT
C
C  OUTPUT: FVAL, OPT
C
C  SUBROUTINES CALLED: SPECIFIED BY USERS
C
C  DESCRIPTION: CKTANALYSIS IS A SUBROUTINE WHICH IS WRITTEN BY THE
C               USER.  ITS PURPOSE IS TO CARRY OUT CIRCUIT ANALYSIS
C               AND EVALUATE THE OBJECTIVE FUNCTION.
C  FVAL -- THE VALUE OF THE OBJECTIVE FUNCTION (REAL)
C  X -- THE PARAMETERS TO BE OPTIMIZED (REAL; MAX. DIMENSION IS 24 IN
C       PROGRAM)
C  OPT -- A FLAG USED FOR INDICATING WHETHER FVAL IS NEEDED.  WHEN
C         OPT = 1, FVAL IS NEEDED; WHEN OPT = 0, CARRY OUT NORMAL
C         RCUIT ANALYSIS (INTEGER)
C
```

```
C***********************************************************************
      SUBROUTINE CKTANALYSIS(X,FVAL,OPT)
      IMPLICIT REAL*8 (A-H,L,K,O-Z)
C
C  !!! DIMENSION OF X NEEDS TO BE CHANGED IF MORE THAN 24 PARAMETERS ARE
C       USED
C
C  !!! MORE ELEMENTS CAN BE ADDED HERE
C
      DIMENSION A1(5),B1(5),C1(5),D1(5),E1(5),F1(5),G1(5),H1(5),X(24)
      DOUBLE COMPLEX A(5),B(5),C(5),D(5),E(5),F(5),G(5),H(5)
      INTEGER OPT
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,FREQ,ICOU,DB(101,18)/B1/PI
      ICOU = 0
      IFLAG = 0
C
C  !!! FARANT'S REF ZO IS ASSIGNED ONLY HERE
C
      ZO = 50.D0
C
C
+---------+---------+---------+---------+---------+---------+---------+
C    USER'S PROGRAM BEGIN IN THE FOLLOWING LINES
C
      CLIN = X(1)
      CLFB = ATAND(X(2))/100.D0+1.1D0
      ROUT = 10.D0+EXP(X(3))
      CLOUT = X(4)**2
      FREQ = 1.6D0
      IF (OPT .EQ. 1) GO TO 10
      WRITE(6,15) CLIN,CLFB,ROUT,CLOUT
   15 FORMAT(' LIN = ',G14.7,' LFB = ',G14.7,' ROUT = ',G14.7,' LOUT',
     1' = ',G14.7)
      FREQ = 1.3D0
   12 IF ( FREQ .GT. 1.8D0) THEN
        CALL PRT(4,4)
        RETURN
      END IF
      FREQ = FREQ+0.1D0
   10 CALL RLC(A,A1,'S',1.D0,CLIN,0.D0,'S',300.D0)
      CALL RLC(B,B1,'S',0.D0,0.D0,1.D0,'P',0.D0)
      CALL SOURCE(C,C1,'V','C',40.D0,1.D7,500.D0,0.D0)
      CALL RLC(D,D1,'S',0.D0,0.D0,0.5D0,'P',0.D0)
      CALL RLC(E,E1,'S',0.D0,0.D0,0.06D0,'S',0.D0)
      CALL RLC(F,F1,'S',0.D0,CLFB,0.D0,'P',0.D0)
      CALL RLC(G,G1,'S',ROUT,CLOUT,0.D0,'P',300.D0)
      CALL PAR(C,C1,E,E1)
```

```
      CALL CAS(B,B1,C,C1)
      CALL CAS(B,B1,D,D1)
      CALL NLOAD(B,B1,4,50.D0,70.D0,200.D0/FREQ,3.D0)
      CALL SER(B,B1,F,F1)
      CALL CAS(A,A1,B,B1)
      CALL CAS(A,A1,G,G1)
      CALL SAVECKT(A,A1,4,4,PK)
      CALL NPERFORM(A,A1,1,(50.D0,0.D0),(50.D0,0.D0),GT,TN)
      CALL MTRANS(A,4)
      GS = 25.D0/ABS(A(3))**2
      SN = TN/50.D0
      SM = 10.D0*ABS(A(1))**2
      S = EXP(10.D0*(1.D0-PK))
      FVAL = GS+SN+SM+S
      IF (OPT .EQ. 1) RETURN
      IF ( FREQ .NE. 1.6D0) GO TO 12
      WRITE(6,30) GS,SN,SM,S,FVAL
   30 FORMAT(/' MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6):  ',
     14(2X,G11.4)/' FVALUE = ',G12.5)
      GO TO 12
      END
```

Fig. 6a  The listing of [SW.FARANT1]CIR.FOR.

TYPE "N" TO CHANGE THE SET OF DATA
>
LIN = 15.00000 LFB = 0.4656505 ROUT = 30.08554 LOUT = 25.00000



MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6): 2.257 1.602 6.414 14.13
FVALUE = 24.407

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 1.400 | 0.6650 | 106.8 | 0.0446 | -4.6 | 4.7422 | 59.4 | 0.6055 | -17.7 | 0.61 |
| 1.500 | 0.7423 | 86.3 | 0.0379 | -12.4 | 3.9709 | 48.0 | 0.5929 | -18.7 | 0.67 |
| 1.600 | 0.8008 | 72.3 | 0.0321 | -17.1 | 3.3281 | 39.2 | 0.5919 | -20.0 | 0.74 |
| 1.700 | 0.8432 | 62.3 | 0.0272 | -19.3 | 2.8135 | 32.1 | 0.5965 | -21.6 | 0.81 |
| 1.800 | 0.8739 | 54.8 | 0.0232 | -19.4 | 2.4048 | 26.4 | 0.6033 | -23.6 | 0.89 |

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 | 0.00 |
| 1.500 | 11.98 | 66.92 | 56.07 | 71.80 | -12.15 | 3.01 | 50.00 | 0.00 |
| 1.600 | 10.44 | 80.08 | 56.03 | 71.82 | -30.09 | 3.00 | 50.00 | 0.00 |
| 1.700 | 8.99 | 102.79 | 55.99 | 71.84 | -47.06 | 3.00 | 50.00 | 0.00 |
| 1.800 | 7.62 | 133.60 | 55.96 | 71.87 | -63.20 | 2.99 | 50.00 | 0.00 |

TYPE ANY CHARACTER TO CONTINUE >

INITIAL VALUES OF VARIABLES ARE:

15.00000 -2.000000 3.000000 5.000000

INITIAL FUNCTION VALUE = 24.40722

SENSITIVITIES:
16.608 -61.203 -32.131 65.023

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

STEP # 1 [X]:

14.912 0.44582 3.8560 3.9606
FVAL = 8.596773

SENSITIVITIES:
19.255 -0.10626 1.9733 -2.2912

STEP # 2 [X]:
13.622 0.83821 3.4015 4.4699
FVAL = 6.713297

SENSITIVITIES:
14.750 0.40446E-01 0.45704 -0.23838

STEP # 3 [X]:
8.6000 1.1500 2.8281 4.6855
FVAL = 4.181706

SENSITIVITIES:
-1.4315 0.54916 -0.61487 1.4436

STEP # 4 [X]:
9.3127 0.91939 3.0780 4.4125
FVAL = 3.962469

SENSITIVITIES:
0.54720 0.39076 -0.18670E-01 0.13363

STEP # 5 [X]:
9.0505 0.70756

Fig. 6b Results of the optimization using FARANT 1.0.

```
STEP #   31 [X]:
  9.0832     0.13207        1.5289        3.8613
                                                          FVAL =    3.684124
SENSITIVITY:
-0.13876E-07   0.60408E-09  -0.82622E-09   0.10638E-07

OPTIMIZATION HAS TERMINATED AFTER   32 STEPS.

    INITIAL FVAL =   24.40722          INITIAL VALUES:

      15.00000          -2.000000         3.000000          5.000000

    FINAL FVAL =   3.684124          FINAL VALUES:




      9.083211          0.1320683         1.528931          3.861288

LIN =   9.083211      LFB =   1.175234     ROUT =    14.61324     LOUT =   14.90955

MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6):      1.897       1.498       0.2358       0.5270E-(
FVALUE =   3.6841
```

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 1.400 | 0.2539 | -112.4 | 0.0409 | 97.5 | 3.9817 | 105.1 | 0.7156 | 14.9 | 1.39 |
| 1.500 | 0.1603 | -149.4 | 0.0447 | 96.9 | 3.8316 | 95.1 | 0.7196 | 10.0 | 1.36 |
| 1.600 | 0.1535 | 153.5 | 0.0489 | 96.1 | 3.6301 | 85.6 | 0.7240 | 5.4 | 1.29 |
| 1.700 | 0.2271 | 116.8 | 0.0535 | 95.1 | 3.3970 | 76.8 | 0.7286 | 0.9 | 1.22 |
| 1.800 | 0.3169 | 98.0 | 0.0583 | 93.8 | 3.1507 | 68.7 | 0.7331 | -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 | 12.00 | 115.37 | 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 |

```
TYPE ANY CHARACTER TO CONTINUE >
STATISTICS FOR THIS JOB:
ELAPSED TIME =    10.94141         SEC.
CPU TIME =         9150 MS         BUFFER I/O COUNTED =          0
DIRECT I/O COUNTED =               0     PAGE FAULTS COUNTED=          0
TIME: 16:11:27                                                 DATE: 16-AUG-83
```

```
C*********************************************************************
C
C      THIS IS THE MAIN PROGRAM OF THE FARANT
C
C*********************************************************************
C
C  ASSIGN COMMON DATA BLOCK TO :
C  IFLAG -- INDICATES THE SUCCESS OF AN OPERATION BY HAVING A VALUE
C           ZERO (INTEGER)
C  ZO -- CHARACTERISTIC IMPEDENCE (REAL)
C  F -- FREQUENCY (REAL)
C  ICOU -- INDICATES THE SIZE OF DB (INTEGER)
C  DB -- DATA BASE FOR STORAGE OF DATA TO BE PRINTED OR PLOTTED (REAL)
C
      DOUBLE PRECISION ZO,DB,PI,F
      CHARACTER*8 HMS,DMY*9
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,F,ICOU,DB(101,18)/B1/PI
      PI = 3.14159265358979323 8D0
      CALL FARSTART
      CALL DATE(DMY)
      CALL TIME(HMS)
      WRITE(6,10) HMS,DMY
   10 FORMAT('  TIME: ',A8,49X,'DATE: ',A9)
      STOP 'SUCCESSFUL EXIT                       FARANT VERSION 2.0'
      END
C************************************************************************
C
C  FUNCTION: CKTANALYSIS
C
C  INPUT: X,OPT
C
C  OUTPUT: FVAL, OPT
C
C  SUBROUTINES CALLED: SPECIFIED BY USERS
C
C  DESCRIPTION: CKTANALYSIS IS A SUBROUTINE WHICH IS WRITTEN BY THE
C               USER.  ITS PURPOSE IS TO CARRY OUT CIRCUIT ANALYSIS
C               AND EVALUATE THE OBJECTIVE FUNCTION.
C  FVAL -- THE VALUE OF THE OBJECTIVE FUNCTION (REAL)
C  X -- THE PARAMETERS TO BE OPTIMIZED; ITS MAXIMUN NUMBER IS 24 BUT IT
C       CAN BE MODIFIED (REAL)
C  OPT -- A FLAG USED FOR INDICATING WHETHER FVAL IS NEEDED WHEN
C         OPT = 1, FVAL IS NEEDED; WHEN OPT = 0, CARRY OUT NORMAL
C         CIRCUIT ANALYSIS. (INTEGER)
C
C************************************************************************
      SUBROUTINE CKTANALYSIS(X,FVAL,OPT)
```

```
      IMPLICIT REAL*8 (A-H,L,K,O-Z)
C
C  !!! DIMENSION OF X NEEDS TO BE CHANGED IF MORE THAN 24 PARAMETERS ARE
C       USED
C
      DIMENSION X(24)
C
C  !!! MORE ELEMENTS CAN BE ADDED HERE
C
      DIMENSION A(4,4),B(4,4),C(4,4),D(4,4),E(4,4),F(4,4),G(4,4),H(4,4)
      INTEGER OPT
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,FREQ,ICOU,DB(101,18)/B1/PI
      ICOU = 0
      IFLAG = 0
C
C  !!! FARANT'S REF ZO IS ASSIGNED ONLY HERE
C
      ZO = 50.D0
C
C
+---------+---------+---------+---------+---------+---------+---------+
C    USER'S PROGRAM BEGIN IN THE FOLLOWING LINES
C
      LIN = X(1)
      LFB = ATAND(X(2))/100.D0+1.1D0
      ROUT = 10.D0+EXP(X(3))
      LOUT = X(4)**2
      FREQ = 1.6D0
      IF (OPT .EQ. 1) GO TO 10
      WRITE(6,15) LIN,LFB,ROUT,LOUT
   15 FORMAT(' LIN = ',G14.7,' LFB = ',G14.7,' ROUT = ',G14.7,' LOUT',
     1' = ',G14.7)
      FREQ = 1.3D0
   12 IF ( FREQ .GT. 1.8D0) THEN
        CALL PRT(4,4)
        RETURN
      END IF
      FREQ = FREQ+0.1D0
   10 CALL RLC(A,'S',1.D0,LIN,0.D0,'S',300.D0)
      CALL RLC(B,'S',0.D0,0.D0,1.D0,'P',0.D0)
      CALL SOURCE(C,'V','C',40.D0,1.D7,500.D0,0.D0)
      CALL RLC(D,'S',0.D0,0.D0,0.5D0,'P',0.D0)
      CALL RLC(E,'S',0.D0,0.D0,0.06D0,'S',0.D0)
      CALL RLC(F,'S',0.D0,LFB,0.D0,'P',0.D0)
      CALL RLC(G,'S',ROUT,LOUT,0.D0,'P',300.D0)
      CALL PAR(C,E)
      CALL CAS(B,C)
```

```
      CALL CAS(B,D)
      CALL NLOAD(B,4,50.D0,70.D0,200.D0/FREQ,3.D0)
      CALL SER(B,F)
      CALL CAS(A,B)
      CALL CAS(A,G)
      CALL SAVECKT(A,4,4,PK)
      CALL NPERFORM(A,1,(50.D0,0.D0),(50.D0,0.D0),GT,TN)
      CALL MTRANS(A,4,0)
      GS = 25.D0/(A(2,1)**2+A(2,2)**2)
      SN = TN/50.D0
      SM = 10.D0*(A(1,1)**2+A(1,2)**2)
      S = EXP(10.D0*(1.D0-PK))
      FVAL = GS+SN+SM+S
      IF (OPT .EQ. 1) RETURN
      IF ( FREQ .NE. 1.6D0) GO TO 12
      WRITE(6,30) GS,SN,SM,S,FVAL
   30 FORMAT(/' MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6):  ',
     14(2X,G11.4)/' FVALUE = ',G12.5)
      GO TO 12
      END
```

Fig. 6c The listing of [SW.FARANT2]CIR.FOR

TYPE "Y" TO HAVE OPTIMIZATION
TYPE "N" TO DO NORMAL CIRCUIT ANALYSIS
>
WHAT IS THE NUMBER (INTEGER) OF PARAMETERS TO BE OPTIMIZED ?
PLEASE ENTER THE INITIAL GUESSES OF:
CAUTION: USE NO ZEROS
PARAMETERS # 1 = ?
PARAMETERS # 2 = ?
PARAMETERS # 3 = ?
PARAMETERS # 4 = ?
DATA ENTERED ARE AS FOLLOWS:
15.00000 -2.000000 3.000000 5.000000
TYPE "Y" IF DATA ARE CORRECT
TYPE "N" TO CHANGE THE SET OF DATA
>
LIN = 15.00000 LFB = 0.4656505 ROUT = 30.08554 LOUT = 25.00000

MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6): 2.257 1.602 6.414 14.1
FVALUE = 24.407

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 1.400 | 0.6650 | 106.8 | 0.0446 | -4.6 | 4.7422 | 59.4 | 0.6055 | -17.7 | 0.61 |
| 1.500 | 0.7423 | 86.3 | 0.0379 | -12.4 | 3.9709 | 48.0 | 0.5929 | -18.7 | 0.67 |
| 1.600 | 0.8008 | 72.3 | 0.0321 | -17.1 | 3.3281 | 39.2 | 0.5919 | -20.0 | 0.74 |
| 1.700 | 0.8432 | 62.3 | 0.0272 | -19.3 | 2.8135 | 32.1 | 0.5965 | -21.6 | 0.81 |
| 1.800 | 0.8739 | 54.8 | 0.0232 | -19.4 | 2.4048 | 26.4 | 0.6033 | -23.6 | 0.89 |

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 | 0.00 |
| 1.500 | 11.98 | 66.92 | 56.07 | 71.80 | -12.15 | 3.01 | 50.00 | 0.00 |
| 1.600 | 10.44 | 80.08 | 56.03 | 71.82 | -30.09 | 3.00 | 50.00 | 0.00 |
| 1.700 | 8.99 | 102.79 | 55.99 | 71.84 | -47.06 | 3.00 | 50.00 | 0.00 |
| 1.800 | 7.62 | 133.60 | 55.96 | 71.87 | -63.20 | 2.99 | 50.00 | 0.00 |

TYPE ANY CHARACTER TO CONTINUE >

INITIAL VALUES OF VARIABLES ARE:

15.00000 -2.000000 3.000000 5.000000

INITIAL FUNCTION VALUE = 24.40722

SENSITIVITIES:
16.608 -61.203 -32.131 65.023

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

STEP # 1 [X]:

14.912 0.44582 3.8560 3.9606
FVAL = 8.596773
SENSITIVITIES:
19.255 -0.10626 1.9733 -2.2912

STEP # 2 [X]:
13.622 0.83821 3.4015 4.4699
FVAL = 6.713297
SENSITIVITIES:
14.750 0.40446E-01 0.45704 -0.23838

SENSITIVITY:
-0.58163E-05 0.16583E-06 0.87022E-08 0.72880E-07

STEP # 31 [X]:
9.0832 0.13207 1.5289 3.8613
FVAL = 3.684124

SENSITIVITY:
-0.13809E-07 0.60426E-09 -0.83046E-09 0.10632E-07

OPTIMIZATION HAS TERMINATED AFTER 32 STEPS.

INITIAL FVAL = 24.40722 INITIAL VALUES:

15.00000 -2.000000 3.000000 5.000000

FINAL FVAL = 3.684124 FINAL VALUES:

9.083211 0.1320683 1.528931 3.861288

,IN = 9.083211 LFB = 1.175234 ROUT = 14.61324 LOUT = 14.90955

MEASURES FOR GAIN, NOISE, MATCH, K-FACT (F=1.6): 1.897 1.498 0.2358 0.5270E-01
VALUE = 3.6841

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 1.400 | 0.2539 | -112.4 | 0.0409 | 97.5 | 3.9817 | 105.1 | 0.7156 | 14.9 | 1.39 |
| 1.500 | 0.1603 | -149.4 | 0.0447 | 96.9 | 3.8316 | 95.1 | 0.7196 | 10.0 | 1.36 |
| 1.600 | 0.1535 | 153.5 | 0.0489 | 96.1 | 3.6301 | 85.6 | 0.7240 | 5.4 | 1.29 |
| 1.700 | 0.2271 | 116.8 | 0.0535 | 95.1 | 3.3970 | 76.8 | 0.7286 | 0.9 | 1.22 |
| 1.800 | 0.3169 | 98.0 | 0.0583 | 93.8 | 3.1507 | 68.7 | 0.7331 | -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 | 12.00 | 115.37 | 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 |

TYPE ANY CHARACTER TO CONTINUE >
STATISTICS FOR THIS JOB:
ELAPSED TIME = 10.44922 SEC.
CPU TIME = 9960 MS BUFFER I/O COUNTED = 0
DIRECT I/O COUNTED = 0 PAGE FAULTS COUNTED= 0
TIME: 16:14:00 DATE: 16-AUG-83

Fig. Results of the optimization using FARANT 2.0.

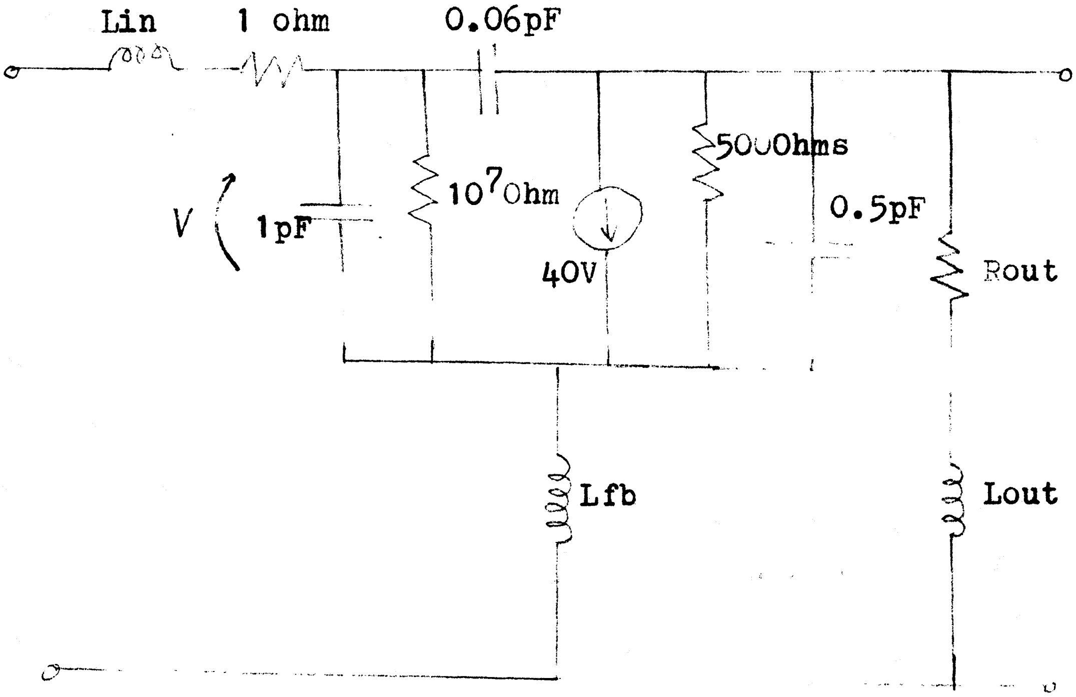



Fig. 6e Circuit diagram of the network to be optimized.

In the printout the following significent differences and similarities between **BASIC** and **FORTRAN** version of **FARANT** can be found:

1. In the **FORTRAN** program, all the variable types have to be declared at the beginning of the program.

2. In the **FORTRAN** versions, every real constant must be represented in exponential form using "D" instead of "E", e.g. 1 as real constant should be written as 1.D0

3. All the program statements are typed in upper case for the **FORTRAN** versions.

4. In the **FORTRAN** versions, FREQ instead of F is used to represent frequency in the subroutine CKTANALYSIS. However, F is used for representing frequency in the other subroutines of **FARANT.**

5. IF and GOTO statements are used in **FORTRAN** program instead of FOR and NEXT statements.

6. END and RETURN replace SUBEND and SUBEXIT in **FORTRAN** subroutines.

7. **FORTRAN** does not have OPTION BASE, DEG and FIXED statements.

8. ICOU and IFLAG substitute for COUNT and NOGO in **FARANT 1.0** and **2.0**.

9. The square of any variable, X, is represented by X**2 in FORTRAN.

10. FORMAT statement is used in **FORTRAN** to put the printout in right format.

11. All the results of **BASIC** and **FORTRAN** versions agree up to 4 digits.

12. **FARANT 2.0** and **.10** only took about 9.2 and 10.6 seconds of cpu time to obtain the answers while **BASIC** version took about 10 minutes to obtain the same results. However, this is not a fair comparison because the **HP 9845** compiles, executes and prints results at the same time but **FARANT** in the **VAX** has been compiled before it is run. Moreover, the cpu time shown for **FORTRAN** versions does not account for the time spent in compiling CKTANALYSIS.

13. Both **BASIC** and **FORTRAN** versions took about 30 steps to optimize the function.

14. At the end of parameter printing, the terminal screen will be frozen for the **FORTRAN** versions until the carriage return is hit.

Differences on FARANT 1.0 and 2.0 observed from the results and programs are:

1. **FARANT 1.0** is faster than **FARANT 2.0** by one second cpu time because the two port matrices in **FARANT 2.0** have to be rearranged. (For details, see section 3.3)

2. Results of both **FORTRAN** versions are almost the same except the sensitivies are different. This is owing to round-off error in calculation of gradient. The details about the

calculation of gradient will be found in section 3.4.1.

3. In subroutine CKTANALYSIS the two-port descriptions of **FARANT 2.0** contain in (4x4) arrays A through H but those of **FARANT 1.0** contain in (5X1) real matrices A1 through H1 and (5X1) complex matrices A through H.

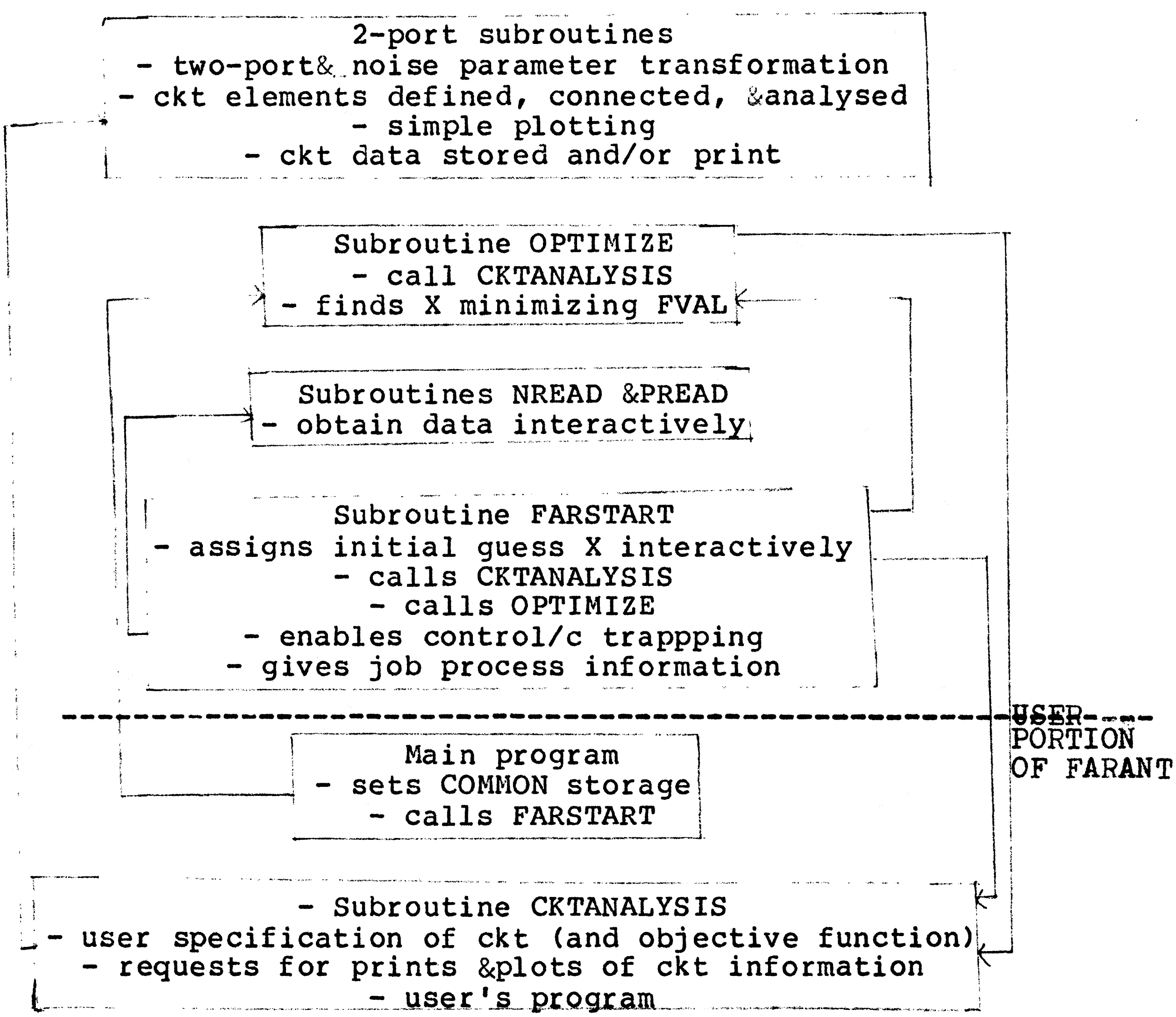
4. In **FARANT 1.0**, magnitude of S parameters are computed using the function ABS(X(I)) but calculated in **FARANT 2.0** using the formula:

$$|X(I)| = [(\text{real part of } X(I))^2 + (\text{imaginary part of } X(I))^2]^{1/2}$$

## 3.0 PROGRAMMING POINT OF VIEW AND SOME SUGGESTIONS

This section provides a summary of the differences between **BASIC** and **FORTRAN** versions, differences between **FARANT 1.0** and **2.0**, and differences between **BASIC** language in **HP 9845** and **VAX FORTRAN.** Suggestions are given in both programming and the setup of **Digital Command Language** (DCL) procedure in the **VAX.** Finally, an example will be given to illustated these suggestions.

### 3.1 Flow Chart of FARANT



## 3.2 Initialization in The Subroutine CKTANALYSIS

The user is advised to copy all the DATA and COMMON statements in the subroutine CKTANALYSIS exactly, if he wants to use his own subroutines in **FARANT.** In the following paragraphs, statements apply only to **FARANT 1.0** is enclosed by braces ,{}, while statements apply only to **FARANT 2.0** is enclosed by bracket,[].

```
IMPLICIT REAL*8 (A-H,L,K,O-Z)
```

This statement declares all the variables begin with letter A through H, O through Z, L and K are real variable with double precision.

```
[DIMENSION X(24)]
[DIMENSION A(4,4),B(4,4),...,H(4,4)]
{DIMENSION A1(5),B1(5),...H1(5)}
{DOUBLE COMPLEX A(5),B(5),...H(5)}
```

These statements allocate space to store the two-port descriptions and objective variables for OPTIMIZE. In **FARANT 1.0**, all the descriptions for a two-port are stored in one (4X4) real matrix. In **FARANT 2.0**, all the descriptions for a two-port are stored in a (5X1) complex matrix for two-port parameters and a (5X1) real matrix for noise parameters. The number of matrices for two-port descriptions and the dimension of objective variables can be increased.

INTEGER OPT

This statement declares OPT is an integer variable, otherwise OPT will be a real variable. The user needs not copy this statements in his subroutines.

COMMON IFLAG,ZO,FREQ,ICOU,DB(101,18)/B1/PI

This statement puts the above variables in common storage so that subroutines in **FARANT** can have access to these variables.

IFLAG -- It is the same as "NOGO" in **BASIC** and signals the failure in two-port transformation if it is one.

ZO -- It is the reference impedance for S parameters and is defined to be 50 Ohms in the subroutine CKTANALYSIS.

FREQ -- FREQ respresents frequency and is the same as as "F" in **BASIC** version.

ICOU -- It holds the number of rows in data base, DB.

DB(101,18) -- It is the data base in double precision. The user needs not copy it into his subroutines if he does not use DB.

/B1/PI -- PI stored the pi constant in common storage. If the user wants to use pi, he must copy "/B1/PI" to the COMMON statement of his programs.

### 3.3 Complex Number Manipulations in FORTRAN VERSIONS

Using a 2X2 matrix to represent a complex number is one of the reasons that the **BASIC** version is slow. However, **VAX FORTRAN** supports complex number operations. Therefore, most of the matrix operations in **FARANT** are replaced by complex algebra. In **FORTRAN**, complex numbers are represented in rectangular coordinates, e.g. (X,Y) where X and Y are the real and imaginary parts, respectively. It is used extensively in **FARANT 1.0** but not in **FARANT 2.0**; only subroutines ZIO, MTRANS, NTRANS, CAS, LOSSYLINE, and NPERFORM in **FARANT 2.0** use complex algebra. Sometimes, subroutines in **FARANT 2.0** use real numbers or polar coordinates to manipulate complex numbers. The conversion between polar and rectangular coordinates are done by subroutine POLAR. It actually contains only two statements:

```
PH = ATAN2D (X,Y)
MAG = (X*X + Y*Y)
```

The function ATAN2D in the math library of **FORTRAN** will perform the same function as the statements for finding PH in SUB POLAR of the **BASIC** version. To sum up, the complex number manipualtion and accuracy in **FORTRAN** versions heavily depends on the math library in **VAX FORTRAN**.

Since the complex number manipulation in **FARANT 2.0** is quite different from that of **FARANT 1.0**, the argurments for some of the subroutines -- MTRANS, ZIO and NTRANS -- have one more argument ,KFLAG , that does not occur in corresponding subroutines of **FARANT 1.0**. This KFLAG enables subroutines using complex algebra to call the other subroutines using complex algebra directly. For example, ZIO can call MTRANS without transforming complex numbers into real numbers which then are passed to MTRANS and converted back into complex numbers in MTRANS. This problem in calling other subroutines does not occur in subroutines using real numbers to represent complex numbers. Moreover, the two-port descriptions of those subroutines using complex operations in the whole routines need to be rearranged after entering and before leaving the subroutines. As a result, **FARANT 2.0** is a little bit slower than **FARANT 1.0**.

## 3.4 Differences betweeen The BASIC and FORTRAN Versions

Differences that affect the results of **FARANT** are mentioned first. The other differences will be found under the heading miscellaneous.

### 3.4.1 The Calculation of Gradient in The Subroutine OPTIMIZE -

Two subroutines, GRAD and GRADIENT, are used by the subroutine OPTIMZIE for finding the gradient of a objective funtion. Subroutine GRADIENT takes two points of the objective function to find the gradient at one of the two points using the formula:

$$gradient = (f(X+10^{-5}*XINIT)-f(X))/(10^{-5}*XINIT)$$

Where f(X) = value of the objective function at X
XINIT = initial guess of X

The above expression is not accurate since it actually gives the gradient for the point between X and X+1.E-5. The inaccuracy becomes significant when the optimization reaches a relative "flat" region at which the objective function changes very little with the objective variables. As a result, the subroutine OPTIMIZE loses the "sense of direction" and the accurate of final variable values may be limited.

Therefore, the subroutine GRAD is introduced. It computes the gradient using following expression:

$$gradient = (f(X + 10^{-5}*XINIT) - f(X - 10^{-5}*XINIT))/(2*XINIT*10^{-5})$$
$$-(XINTI*10^{-5})^{2}*f(X)^{3}/6$$

This expression uses three points of the objective function to compute a gradient. Consequently, the speed of optimization will be decreased. Moreover, the accuracy of the subroutine GRADIENT becomes important at the last state of optimization but not at the beginning. As a compromise, the subroutine OPTIMIZE will switch to GRAD when the change in FVAL, the value of the objective function, is less than 5E-6. Further, the loss in speed is compensated by the reduction in number of steps and the activity of searching around.

### 3.4.2 **Modifications in Subroutine NPERFORM** -

Subroutine NPERFORM is the same as SUB Nperformance in the **BASIC** version. Some modifications in calcultaion of gain has been made to avoid the error in taking the logarithm of a negative number. In the **BASIC** version, gain is calculated as follows:

GTYPE = 1

$$GT = \frac{4*|ZIN|^2*RS*RL}{|ZIN+ZS|^2*|A*ZL+B|^2}$$

GTYPE = 2

$$Gp = GT|_{ZS = ZIN^*}$$

GTYPE = 3

$$Ga = \frac{RS}{ROUT*|A+C*ZS|^2}$$

GTYPE = 4

$$Gmax = \frac{1}{|A*B-C*D|*(K+(K^2-1))^{1/2}}$$

If the numerator of the expression of GYTPE = 1 through 3 is negative or zero, it is set to 1E -99. If the denominator of the expression equals zero, it is set to 1E 99. By this way, taking logarithm of a negative number is avoided when GTYPE is equal to 1 or 2 but not equal to 3 because ROUT may be negative.

However, the formula for GTYPE = 2 is different in the **FORTRAN** versions.

$$Gp = \frac{|ZIN|^2 * RL}{RIN * |A*ZL+B|^2}$$

If the numerator of the expression for gain is zero or either the numerator or denominator, but not both, of the expression is negative, the numerator is set to 1E -38. If the denominator of the expression equals zero, it is set to 1E 38. Therefore, results of both version are consistent if ROUT is positive. Otherwise, results may not be the same.

### 3.4.3 **Miscellaneous** -

Summary of the small differences between **BASIC** and **FORTRAN** versions are as follows:

1. Subroutines PREAD and NREAD are changed to acquire data interactively for the **FORTRAN** versions(For details, see sections 2.2.1.6 and 2.2.1.7)

2. In **FARANT 2.0**, MTRANS, NTRANS and ZIO have one more parameter, KFLAG. (For details, see sections 3.3)

3. Subroutine FARSTART has changed to request data for optimization interactively, enables the trapping of control/c and issues job process information in the **FORTRAN** versions.

4. After printing parameters, the **FORTRAN** versions of subroutine PRT will freeze the screen until the user hits the carriage return.

5. Defined constant like pi is not supported by **FORTRAN.** However, pi is defined in the main program of **FARANT**, the user can use it in his subroutines by adding "/B1/PI" in the COMMON statement.

6. In **FORTRAN** versions, the two-port and noise parameters are arranged differently. Two-port and noise parameters are stored in a real (5X1) and a complex (5X1) arrays respectively for **FARANT 1.0**. For **FARANT 2.0**, a real (4X4) array contians all the parameters. (For details, see section 2.0)

## 3.5 HP 9845 BASIC And VAX FORTRAN

The following list only shows some of the significant differences. It is assumed the user understands **FORTRAN IV.** For detailed information about **VAX FORTRAN,** the user should consult [7].

1. In order to use double precision for all real variables, the IMPLICIT statement has to be used in **FORTRAN** program whereas **HP 9845** always uses 12 digits to represent real variable if the user does not specify.

2. The data type does not assoicate with the first letter of variable in **HP 9845.** However, all variables begin with letter A through H, O through Z, L and K are assumed to be real type with double precision as they are declared in the subroutine CKTANALYSIS of **FORTRAN** versions. Variables begin with other letter are assumed to integer type. Moreover, all the constants have to be written in exponential form, e.g. 1 is 1.D0. The decmial point and exponent are especially <u>important</u> for optimization because contants without them are considered to real numbers with single precision. If the user wants to do optimization, every constant and variable must be in double precision.

3. Apostrophes, ' ', is used to specify a string in **FORTRAN** versions whereas quotes, " ", are used for the same purpose in **HP 9845**, e.g. "S" is used instead of 'S'.

4. Every string variable in **HP 9845** must be followed by a dollar sign, $. On the other hand, names of string variables are the same as for ordinary variables in **FORTRAN** versions.

5. All the local variables within a called program will be saved after the control is returned to the calling program in **VAX FORTRAN.** Hence, the pass parameters can be equated to some local variables during the first call of a subroutine. In the next call, parameters are set to the old values if the user sets a flag inside the subroutine interactively or through the subroutine CKTANALYSIS. This special feature is used in subroutines NREAD and PREAD and saves time for reading data if those data are the same as the previous set. For **HP 9845,** all the variables are set to zero at the beginning each call. The only way to save pass paremeters is to put them in COM statement.

6. Upper case letters will not be translated to lower case in **FORTRAN.** Moreover, upper and lower case letters have different ASCII values. Since **FORTRAN** versions are typed in upper case, the user is advised to lock the keyboard to type capital letters. However, the **BASIC** program is typed in upper and lower cases.

7. In **BASIC** version, the dimension of an array can be passed using the symbol, (*). However, the dimension of an array must be defined in a subroutine or it should be a pass parameter explicitly in **FORTRAN.** Moreover, only the array name is needed to be typed in a CALL statement. For instance, CALL OPTIMIZE(X,N) instead of CALL OPTIMIZE(X(*),N) is used in **FORTRAN** versions.

8. The dimension of an array in **VAX FORTRAN** has a lower bound of one if it is not specified whereas a OPTION BASE statement must be used in **HP 9845.**

9. **FORTRAN** supports complex number operations but the **BASIC** in **HP 9845** does not.

10. However, **VAX FORTRAN** does not have matrix operations which are supported by **HP 9845 BASIC.** The user must write his own subroutines to manipulate matrices.

11. In **HP 9845,** the same name can be used for a single variable and an array but this is invalid in **FORTRAN.** For instance, F for frequency and F(6,4) for two-port descriptions are used in the SUB CKTANALYSIS of **BASIC** version. However, frequency is represented by FREQ in CKTANALYSIS of the **FORTRAN** versions.

12. DEG and RAD statements do not exist in **VAX FORTRAN.** Nevertheless, it does support trigonometric functions in degree and rad. Those functions accept degree as input ususlly have letter "D" at the end of their function name. A good summary of those functions can be found in table C-1 of [7].

13. Since the dimension of an array must be defined within the subroutine, some of the data type statements which are not in the **BASIC** version must be changed in **FORTRAN** version if the user wants to increase the data base, number of objective variable for optimization, and number of two-port identifiers. Those statements need to be changed are indicated by "!!!" in the comment. Therefore, the user can easily find them.

14. The comment line in **VAX FORTRAN** can be the same as **BASIC,** using "!" in the first column. "C" and "D" can also be used as comment line indicators. Actually, "D" is the debug statement indicator. If "/D_LINE" is specified with "FOR" command, e.g. "FOR/D_LINE file name, the compiler will compile those statements with the debug statement indicator. Otherwise, "D" has the same function as "C" or "!".

15. **FORTRAN** program must begin in the sixth column and end in seventy second column.

16. BLOCK IF statement which has the form:

```
IF (expression) THEN
block
ELSE IF (espression) THEN
.
.
.
ELSE
END IF
```

and is not supported by **HP 9845 BASIC** and **FORTRAN IV** can be found in **VAX FORTRAN.** Some other statements that cannot be found in both **HP 9845 BASIC** and **FORTRAN IV** are -- END DO, DO WHILE and PARAMETER statements. The form of DO WHILE statment is as follows:

```
DO s WHILE (expression)
```

where s is the label of the last statement included within the range of a DO statement. The purpose of DO END statement is to terminate the range of a DO statement. For instance, the above example can be written as follows:

```
DO WHILE (expression)
```

.

.

END DO

PARAMETER statement assigns a symbolic name to a constant. Its form is:

PARAMETER (p=c, q=d, ...)

where p and q are the symbolic names and c and d are the contants. The statment is non-executable. The symbolic names are replaced by corresponding contants during compilation.

## 3.6 Suggest Setup in User's Working Areas

In order to make the manipulation of files in **VAX** easier, some command files are set up, besides RUNFRT.COM, under the directory [SW]. They are listed and described in the following paragraphs.

| Name.type | Function |
|---|---|
| LOGIN.COM | Every times the user gets on the **VAX**. It will search for the command file LOGIN.COM. If this file is found, it will be executed. The first command in LOGIN.COM informs the **VAX** user's terminal is VT52 (equivalent to VISUAL 50). This |

command should be changed "SET TER/DEV=VT100" if the user uses a VISUAL 100 terminal. This command also enables the user to get into the keypad mode if he uses the EDT editor. Then the procedure defines symbols "ED" and "C" and executes DIR.COM. LOGIN.COM is shown in the following lines.

```
$set ter/dev=vt52
$run [larry]cookie
$ED:==EDIT/EDT/COMMAND=[SW]COM.EDT
$C:==@[SW]
$@DIR
```

DIR.COM  It shows the names of subdirectories inside the directory and asks which subdirectory the user wants. If the user wants to use any sub-directory, he can type in the subdirectory name. Otherwise, he can hit the carriage return and stay in his directory automatically. The user must be cautious in typing subdirectory names because this simple procedure does not check the existance of the subdirectory name typed. If the user types a wrong name, he may not find his files and must issue the command:
"SET DEFAULT [username]" to get back to his directory or "'C'DIR to execute DIR.COM again. DIR.COM is listed below.

```
$ DIR/NOTR [SW].DIR
$ INQUIRE SDIR " ENTER FILE NAME TO SELECT SUBDIR; HIT RETURN TO CHOOSE THE
MAIN DIR"
$ IF SDIR .NES. "" THEN SDIR="."+SDIR
$ SET DEF [SW'SDIR']
```

COM.EDT — It is a command file containing commands of the EDT editor. Each times the user issues the command symbol "ED" defined in LOGIN.COM. He will enter the EDT editor, get a copy of FMAIN.FOR and see the last six lines of FMAIN.FOR. COM.EDT is shown below.

```
INCL FMAIN.FOR
TY .-5:.-1 /STAY
```

If the user wants to copy these command files, he should change the "[SW]" in the above listings into "[username]". Setting up in this way, the user can store all the command files in his directory and files with similar function are grouped into one subdirectory. When he want to execute a command file in his directory from a subdirectory by typing "'C'file name". The user must not put a space between the apostrophe and the file name. If he wants to write a new program using **FARANT**, he can enter the subdirectory containing **FARANT 1.0** or **FARANT 2.0** and type "ED file specification". He will enter the EDT editor and get a copy of FMAIN.FOR automatically. Therefore, the user is advised to use the EDT editor. Moreover, it does not contain too many commands, most of which can be used by stroking one or two keys on the keypad.

The user can enter the EDT editor by issuing "EDIT/EDT file specification". Then he will be in the line mode of the EDT editor. Within this mode, he can use the following commands the abbrevaitions of which are enclosed by brace, {}.

{R}EPLACE a:b

{M}OVE a:b c:d

{I}NSERT a:b

{CO}PY a:b

{T}YPE a:b

{C}HANGE

{INC}LUDE file specification

where a:b is the range specification and a and b is the starting and ending lines, respectively.

When the user issues "C" (CHANGE), they will enter the keypad mode, he can use the keypad on the right hand side of the keyboard. Hitting the arrow, "<--", and function keys of VISUAL 50 together and then the arrow, "-->", key, the user can get the HELP instruction for each key. Since the keypad is different for different terminals, it is impossible to describe how to use the keypad. For detail information, the user should consult "VAX-II Text Editing Reference Manual". A good introduction and summary can also be found in the updated version of "VAX/VMS Primer". The keypad for the VISUAL 50 is reproduced in the following diagram which should be found on each terminal.

| | | | |
|---|---|---|---|
| GOLD | HELP | DEL L<br>UND L | |
| APPEND | SPECINS | REPLACE | SECT↓ |
| PAGE | FNDNEXT | DEL W | |
| COMMAND | FIND | UND W | |
| ADVANCE | BACKUP | DEL C | |
| BOTTOM | TOP | UND C | |
| WORD | EOL | CUT | ENTER |
| CHGCASE | DEL EOL | PASTE | |
| LINE | | SELECT | SUBSTI- |
| OPEN LINE | | RESET | TUTE |

Fig. 7 The keypad of the VISUAL 50.

## 3.7 Suggustions for Programming in FORTRAN

Following paragraphs discusses some special features of **VAX FORTRAN.** These features may help users to use **VAX FORTRAN** efficiently. Also they will be demonstrated in the next section.

In **VAX FORTRAN**, the user can open, close and assign a file to a certain logical unit. In **FARANT**, unit 5 and 6 are used for reading and printing respectively. And the control/c trapping routine uses unit 7 to write to a file called FIRR.DAT. The form of CLOSE and OPEN statements are listed as follows:

OPEN (list of parameters ...)

CLOSE (list of parameters...)

Some of the useful parameters are:

UNIT -- It specifies the logical unit to which the file is assigned.

FILE -- It is the name of the file to be open. Default file type is DAT. Therefore, the user only needs to give the file name if he wants to create a data file.

DISP -- Some of the options are "SAVE" and "DELETE". "SAVE" means to keep the file. "DELETE" means the file to be deleted when the CLOSE statement is executed. (see [9] for details)

Another unique feature of **VAX FORTRAN** is the retaining of local variables after the control is transfered back to the calling program. This feature enables the user to recall the previous values of pass parameters by equating the pass parameters to the calling program. Its advantage will become more conspicuous in the example after this section.

Namelist-directed READ statement allows the user to put long list of data in small group inside a file. For instance, data for variables -- A, B, C, D and E -- are read in a program. The user can write his program using namelist- directed READ statement as follows:

```
NAMELIST /G1/A,B,C /G2/D,E
```

```
      .
      .
      .
READ(5,NML=G1)
      .
      .
      .
READ(5,NML=G2)
      .
      .
```

where G1 and G2 are group names.

The data file assigned to logical unit 5 will be as follows:

```
$G1
 A = 1
 B = 2
 C = 3
$END
$G2
 D = 4
 E = 5
$END
```

All the statements in the data file should be start at second column because the first column is supposed to contain carriage-control characters. In this way, the **VAX** will search for the group names G1 and G2 sequentially. Therefore, G1 and G2 must be put in the order in which they appear in the READ statement. However, the variable names inside G1 or G2 can be put in any order. Using namelist-directed READ statement, the user can put all the data for several program in one data file. As long as the data are arranged in the order that the group names appear in READ statement, the right data will be read. (see [10])

Lastly, the INCLUDE statement can be used in **FORTRAN** program. It has a form of "INCLUDE 'file specification(module)'/[NO]LIST". The brackets, "[]", in the above statement indicates optional elements. Using this statement, the user can copy a text or source file into their program. Further, he can set up a text library which can be manipulated in the same way as the object module library. Using INCLUDE statement can save space for storage of source file by putting those statements ocurring many times in a text file. (For details ,see [6])

## 3.8 Example: Fitting A FET Model Using Optimization

On the following pages, listings and printout of **FARANT 1.0** and **2.0** are shown for the optimization program. Some of the subroutines used are stored inside a user's library. Pieces of these programs can be found under the subdirectories [SW.FARANT1] and [SW.FARANT2]. The following files can be found in both subdirectories having the same names.

| File.type | Descriptions |
|---|---|
| FITNE.FOR | It cantains the user's version of CKTANALYSIS for fitting a FET model and the main program of **FARANT.** |
| FITNE.EXE | It is the image of the optimization program for program execution. |

SWLIB.FOR — It is the **FORTRAN** source file containing user's subroutines -- OUTPUT describing the output circuit, INPUT describing the input circuit, FET describing the FET model, PACK containing the two-port descriptions for input and output circuits and UNPACK recalling the stored two-port description of input and output circuits.

SWLIB.OLB — The object module library of SWLIB.FOR.

FIT2AUG83 — It contains the component values of input and output circuits and the NE673A FET model.

The purpose of this program was to fit the FET model of NE673A to the following data:

| Frequency | Square of S parameters | | | |
|---|---|---|---|---|
| (Ghz) | S11 | S12 | S21 | S22 |
| 21.6 | 0.16 | 0.01 | 3.16 | 0.5 |
| 22.6 | 0.06 | 0.025 | 6.00 | 0.008 |
| 23.6 | 0.125 | 0.0125 | 3.16 | 0.4 |

The data were read interactively and stored in array P of the subroutine CKTANALYSIS. However, the data for subroutines INPUT, OUTPUT and FET were read form FIT2AUG83 using the namelist-directed READ statement. The program minimizes the differences between the S parameters calculated by the program and the square of S parameters, P, read using following equation:

$$FVAL = \sum_{f=21.6}^{23.6} 10ER_f(1,1) + ER_f(1,2) + ER_f(2,1) + 10ER_f(2,2)$$

where f = frequency in Ghz

$$ER_f(i,j) = [S_f(i,j)^2 - P_f(i,j)]^2$$

i and j are the subscripts for S parameters e.g. S(1,2)=S12.

The circuits described in subroutines INPUT, OUTPUT and FET are shown in Fig. 13, Fig. 14 and Fig. 15, respectively. As shown in Fig. 15, elements TR(1), TR(3), TR(11), TR(13) and TR(10) of the FET model were going to be optimized. Their initial values were:

| **Variables** | **Initial values** | **Constraining Function** | **Initial X(i)** |
|---|---|---|---|
| TR(1) | 7.3008 Ohms | $X(5)^2$ | 2.702 |
| TR(3) | 0.0001 pF | $X(1)^2$ | 0.01 |
| TR(11) | 0.2933 nH | $X(2)^2$ | 0.5416 |
| TR(6) | 0.5671 µH | $X(3)^2$ | 0.024 |
| TR(13) | 0.30647 nH | $X(4)^2$ | 0.5836 |
| TR(10) | 1.6384 Ohms | $X(6)^2$ | 1.28 |

```
C*********************************************************************
C
C      THIS IS THE MAIN PROGRAM OF THE FARANT
C
C*********************************************************************
C
C  ASSIGN COMMON DATA BLOCK TO :
C  IFLAG -- INDICATES THE SUCCESS OF AN OPERATION BY HAVING A VALUE
C           ZERO (INTEGER)
C  ZO -- CHARACTERISTIC IMPEDENCE (REAL)
```

```
C  F -- FREQUENCY (REAL)
C  ICOU -- INDICATES THE SIZE OF DB (INTEGER)
C  DB -- DATA BASE FOR STORAGE OF DATA TO BE PRINTED OR PLOTTED (REAL)
C
      DOUBLE PRECISION ZO,DB,PI,F
      CHARACTER*8 HMS,DMY*9
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,F,ICOU,DB(101,18)/B1/PI
      PI = 3.14159265358979323 8D0
      CALL FARSTART
      CALL DATE(DMY)
      CALL TIME(HMS)
      WRITE(6,10) HMS,DMY
   10 FORMAT(' TIME: ',A8,49X,'DATE: ',A9)
      STOP 'SUCCESSFUL EXIT                        FARANT VERSION 1.0'
      END
C*********************************************************************
C
C  FUNCTION: CKTANALYSIS
C
C  INPUT: X,OPT
C
C  OUTPUT: FVAL, OPT
C
C  SUBROUTINES CALLED: SPECIFIED BY USERS
C
C  DESCRIPTION: CKTANALYSIS IS A SUBROUTINE WHICH IS WRITTEN BY THE
C               USER.  ITS PURPOSE IS TO CARRY OUT CIRCUIT ANALYSIS
C               AND EVALUATE THE OBJECTIVE FUNCTION.
C  FVAL -- THE VALUE OF THE OBJECTIVE FUNCTION (REAL)
C  X -- THE PARAMETERS TO BE OPTIMIZED (REAL; MAX. DIMENSION IS 24 IN
C      PROGRAM)
C  OPT -- A FLAG USED FOR INDICATING WHETHER FVAL IS NEEDED.  WHEN
C         OPT = 1, FVAL IS NEEDED; WHEN OPT = 0, CARRY OUT NORMAL
C         RCUIT ANALYSIS (INTEGER)
C
C*********************************************************************
      SUBROUTINE CKTANALYSIS(X,FVAL,OPT)
      IMPLICIT REAL*8 (A-H,L,O-Z)
C
C  !!! DIMENSION OF X NEEDS TO BE CHANGED IF MORE THAN 24 PARAMETERS ARE
C      USED
C
C  !!! MORE ELEMENTS CAN BE ADDED HERE
C
      DIMENSION A1(5),B1(5),C1(5),D1(5),E1(5),F1(5),G1(5),H1(5),X(24)
      DIMENSION Q(2,3,12),P(3,4),TR(20),CIN(15),OUT(15),ER(4),X1(70)
      DIMENSION Y1(4,70)
```

```
      NAMELIST /INNET/ CIN /OUTNET/ OUT /NE673A/ TR
      DOUBLE COMPLEX A(5),B(5),C(5),D(5),E(5),F(5),G(5),H(5)
      CHARACTER ANS,CH(12)*18,VAS*21/'X AXIS DB'/,HAS*21/'Y AXIS FREQ'/
      CHARACTER TITL*73/'  1 - S11, 2 - S12, 3 - S21, 4 -S22'/
      CHARACTER CHA(4)/'1','2','3','4'/, NOF*20
      INTEGER OPT
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,FREQ,ICOU,DB(101,18)/B1/PI
      ICOU = 0
      IFLAG = 0
C
C  !!! FARANT'S REF ZO IS ASSIGNED ONLY HERE
C
      ZO = 50.D0
C
C
+---------+---------+---------+---------+---------+---------+---------+
C    USER'S PROGRAM BEGIN IN THE FOLLOWING LINES
C
      IF (IR .EQ. 0) THEN
      FREQ1 = 21.6D0
      F2 = 23.6D0
      NF = 3
      DF = (F2-FREQ1)/(FLOAT(NF)-1.D0)
      WRITE(6,30)
   30 FORMAT('1 PLEASE TYPE THE SQUARE OF S PARAMETERS'/'  THE S ',
     1'PARAMETERS SHOULD BE ENTERED ON THE SAME LINE AS FOLLOWS:')
      DO 40 I = 1,NF
      FT = DF*(I-1)+ FREQ1
   32 WRITE(6,35) FT,I,I,I,I
   35 FORMAT('  AT ',G14.7,' GHz'/7X,'P(',I1,',1),',10X,'P(',I1,',2),',
     110X,'P(',I1,',3),',10X,'P(',I1,',4),')
      READ(5,*) (P(I,J),J = 1,4)
      WRITE(6,50) (I,P(I,J),J = 1,4)
   50 FORMAT('      P(',I1,',1) = ',G14.7,'     P(',I1,',2) = ',G14.7/
     1'      P(',I1,',3) = ',G14.7,'     P(',I1,',4) = ',G14.7/'  TYPE'
     1,' "Y" (IF DATA IS CORRECT) OR "N" (TO CHANGE DATA)'/' > ',$)
      READ(5,'(A)') ANS
      IF (ANS .NE. 'Y') GO TO 32
   40 CONTINUE
      TYPE *,' ENTER NAME OF THE FILE CONTAINING COMPONENT VALUES: '
      READ(5,'(A20)') NOF
      OPEN (UNIT=7,STATUS='OLD',FILE=NOF)
      READ(7,NML=INNET)
      READ(7,NML=OUTNET)
      READ(7,NML=NE673A)
      CLOSE(UNIT=7,DISP='KEEP')
      END IF
```

```
      IF (OPT .EQ. 0) THEN
        DO I = 1,NF
        FREQ = FREQ1 + (FLOAT(I) - 1.D0)*DF
        CALL INPUT(A,A1,CIN)
        CALL PACK(A,A1,Q,1,I)
        CALL OUTPUT(A,A1,OUT)
        CALL PACK(A,A1,Q,2,I)
        END DO
      END IF
      FVAL = 0.D0
      ICOU = 0
      TR(1) = X(5)*X(5)
      TR(3) = X(1)*X(1)
      TR(11) = X(2)*X(2)
      TR(6) = X(3)*X(3)
      TR(13) = X(4)*X(4)
      TR(10) = X(6)*X(6)
      DO 90 I = 1,NF
      FREQ = FREQ1 + (FLOAT(I) - 1.D0)*DF
      CALL UNPACK(G,G1,Q,1,I)
      CALL FET(B,B1,TR,0)
      CALL RLC(A,A1,'P',0.D0,CIN(15),0.D0,'P',0.D0)
      CALL SER(A,A1,B,B1)
      CALL CAS(G,G1,A,A1)
      CALL UNPACK(A,A1,Q,2,I)
      CALL CAS(G,G1,A,A1)
      CALL MTRANS(G,4)
      ER(1) = (ABS(G(1))**2-P(I,1))**2
      ER(2) = (ABS(G(2))**2-P(I,2))**2
      ER(3) = (ABS(G(3))**2-P(I,3))**2
      ER(4) = (ABS(G(4))**2-P(I,4))**2
      FVAL = FVAL + 10.D0*ER(1) + ER(2) + ER(3) + 10.D0*ER(4)
   90 CALL SAVECKT(G,G1,4,0,-1.D0)
      IF (OPT .EQ. 1) RETURN
      WRITE(6,'(1H1)')
   95 DO 100 I = 1,ICOU
      X1(I) = DB(I,1)
      Y1(1,I) = 10.D0*LOG10(DB(I,2)**2+DB(I,3)**2)
      Y1(2,I) = 10.D0*LOG10(DB(I,4)**2+DB(I,5)**2)
      Y1(3,I) = 10.D0*LOG10(DB(I,6)**2+DB(I,7)**2)-30.D0
  100 Y1(4,I) = 10.D0*LOG10(DB(I,8)**2+DB(I,9)**2)
      CALL PLOT(X1,Y1,CHA,1.D0,1.D0,-30.D0,0.D0,0,4,ICOU,VAS,HAS,TITL)
      IF (IR .EQ. 1) THEN
        WRITE(6,101)
  101 FORMAT('  COMPONENT VALUES OF THE INPUT CIRCUIT:'/T30,'ZO',15X,
     1'LENGTH'/1X,60('-'))
        WRITE(6,102) CIN
  102 FORMAT(' INPUT TRANSFORMER',T25,G14.7,5X,G14.7/' TUNING L',T25,
     1G14.7,5X,G14.7/' GATE LEAD',T25,G14.7,5X,G14.7/' GATE BIAS',T25,
     1G14.7,5X,G14.7/' GATE CAP',T25,G14.7,5X,G14.7//' DISCONT. CAP = ',
```

```
   1G14.7,10X,'END CAP = ',G14.7/' BIAS RESISTOR = ',G14.7,10X,'BIAS'
   1,' CAP = ',G14.7/' SOURCE INDUCTOR = ',G14.7)
      WRITE(6,103)
103 FORMAT(/'  COMPONENT VALUES OF THE OUTPUT CIRCUIT:'/T30,'ZO',15X,
   1'LENGTH'/1X,60('-'))
      WRITE(6,104) (OUT(II),II=1,6),(OUT(II),II=10,13),(OUT(II),II=7,
   1  9),OUT(14),OUT(15)
104 FORMAT(' DRAIN LEAD',T25,G14.7,5X,G14.7/' DRAIN BIAS',T25,G14.7,
   15X,G14.7/' DRAIN CAP',T25,G14.7,5X,G14.7/' T2',T25,G14.7,5X,G14.7
   1/' OUT LINE',T25,G14.7,5X,G14.7//' END CAP = ',G14.7,10X,'DISCON.'
   1,' CAP = ',G14.7/' DISCON. CAP = ',G14.7,10X,'BIAS BYPASS CAP ',
   1'= ',G14.7/' BIAS BYPASS RESISTOR = ',G14.7)
    END IF
    CALL PRT(4,-4)
    WRITE(6,105) TR(1),TR(3),TR(11),TR(6),TR(13),TR(10)
105 FORMAT(2X,'TR(1) = ',G14.7,' TR(3) = ',G14.7,' TR(11) = ',
   1G14.7/'  TR(6) = ',G14.7,' TR(13) = ',G14.7,' TR(10) = ',G14.7)
    IR = 1
    RETURN
    END
```

Fig. 8 Program listing of [SW.FARANT1]FITNE.FOR.

```
X     0|                                                                 Y
  -15004
A -30001                                                                 A
X -45001                                 4                             X X
I -60001                                                                 I
S -75001                                                                 S
  -90001                                 1
D-105001                                                                 F
B-120001                                                                 R
 -135001                                                                 E
 -150001                                                                 Q
 -165001
 -180001
 -195001                                 2
 -210001                                                               2
 -225001
 -240001                                 3
 -255001                                                               3
 -270002
 -285001
 -300003---------+---------+---------+---------+---------+---------+---------+
E -3  21600    21886    22171    22457    22743    23029    23314    23600
  1 - S11, 2 - S12, 3 - S21, 4 -S22                                   FITE -3
 TYPE ANY CHARACTER TO CONTINUE. >
```

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 21.600 | 0.6775 | 160.5 | 0.0455 | 95.7 | 0.9944 | 95.6 | 0.9073 | 158.5 | 0.00 |
| 22.600 | 0.3375 | 124.9 | 0.1020 | 40.8 | 2.0365 | 40.8 | 0.6444 | 138.3 | 0.00 |
| 23.600 | 0.6010 | -133.2 | 0.0959 | -57.8 | 1.7537 | -57.8 | 0.5840 | -161.7 | 0.00 |

```
TYPE ANY CHARACTER TO CONTINUE >
TR(1) =    7.300804    TR(3) =   0.1000000E-03 TR(11) =   0.2933306
TR(6) =  0.5760000E-03 TR(13) =   0.3064730     TR(10) =    1.638400

 INITIAL VALUES OF VARIABLES ARE:

   0.1000000E-01    0.5416000        0.2400000E-01    0.5536000

    2.702000          1.280000

 INITIAL FUNCTION VALUE =    12.37276

SENSITIVITIES:
-0.20402        -141.59         0.91065        -426.78         49.667
  13.835


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


STEP #      1 [X]:
 0.11518E-01   0.56106        0.21176E-01   0.61098         2.7006
  1.2792
                                                        FVAL =    42.60838

 -0.12267E-03

 STEP #    82 [X]:
 -0.44503E-01   0.53734        0.55898E-01   0.56312         2.6484
   1.0853
                                                        FVAL =    7.269432
 SENSITIVITY:
 -0.36338E-05   0.43594E-03  -0.88313E-05   0.57542E-03  -0.66051E-04
 -0.31946E-04

 STEP #    83 [X]:
 -0.44502E-01   0.53734        0.55897E-01   0.56312         2.6484
   1.0853
                                                        FVAL =    7.269432
 SENSITIVITY:
  0.71246E-07   0.12559E-05  -0.96863E-07   0.12006E-05  -0.18494E-05
 -0.42770E-06


 OPTIMIZATION HAS TERMINATED AFTER   84 STEPS.
```

```
X     0|                                                                       Y
  -15004
A -30001                                                                     4 A
X -45001                                                                     1 X
I -60001                                                                       I
S -75001                                    4                                  S
  -90001                                    1
D-105001                                                                       F
B-120001                                                                       R
 -135001                                                                       E
 -150001                                                                       Q
 -165001
 -180001                                    2
 -195001
 -210001                                                                     2
 -225001                                    3
 -240001
 -255002                                                                     3
 -270001
 -285003
 -30000+----------+----------+----------+----------+----------+----------+----------+
E -3  21600   21886   22171   22457   22743   23029   23314   23600
  1 - S11, 2 - S12, 3 - S21, 4 -S22                                      FITE -3
```

TYPE ANY CHARACTER TO CONTINUE. >

COMPONENT VALUES OF THE INPUT CIRCUIT:

| | ZO | LENGTH |
|---|---|---|
| INPUT TRANSFORMER | 26.00000 | 0.1260000 |
| TUNING L | 50.00000 | 0.1450000 |
| GATE LEAD | 100.0000 | 0.1000000E-01 |
| GATE BIAS | 90.00000 | 0.1250000 |
| GATE CAP | 39.00000 | 0.7500000E-01 |

DISCONT. CAP = 0.0000000E+00 END CAP = 0.4000000E-01
BIAS RESISTOR = 50.00000 BIAS CAP = 0.5000000
SOURCE INDUCTOR = 0.1000000E-02

COMPONENT VALUES OF THE OUTPUT CIRCUIT:

| | ZO | LENGTH |
|---|---|---|
| DRAIN LEAD | 100.0000 | 0.1000000E-01 |
| DRAIN BIAS | 90.00000 | 0.1250000 |
| DRAIN CAP | 39.00000 | 0.4000000E-01 |
| T2 | 15.00000 | 0.1260000 |
| OUT LINE | 50.00000 | 0.0000000E+00 |

END CAP = 0.4000000E-01 DISCON. CAP = 0.3000000E-01
DISCON. CAP = 0.3000000E-01 BIAS BYPASS CAP = 0.5000000
BIAS BYPASS RESISTOR = 50.00000

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 21.600 | 0.7209 | 160.1 | 0.0542 | 98.3 | 1.1384 | 92.2 | 0.8742 | 156.6 | 0.00 |
| 22.600 | 0.3611 | 106.3 | 0.1289 | 34.2 | 2.4704 | 27.4 | 0.4469 | 135.8 | 0.00 |
| 23.600 | 0.6029 | -135.2 | 0.0887 | -64.0 | 1.5565 | -71.6 | 0.7496 | -163.5 | 0.00 |

TYPE ANY CHARACTER TO CONTINUE >
TR(1) = 7.014131 TR(3) = 0.1980460E-02 TR(11) = 0.2887337

TR(6) = 0.3124524E-02 TR(13) = 0.3171046 TR(10) = 1.177848
STATISTICS FOR THIS JOB:
ELAPSED TIME = 72.83203 SEC.
CPU TIME = 63440 MS BUFFER I/O COUNTED = 0
DIRECT I/O COUNTED = 0 PAGE FAULTS COUNTED= 0
TIME: 16:03:14 DATE: 16-AUG-83

Fig. 9 Results of the optimization program using FARANT 1.0

```
C*********************************************************************
C
C      THIS IS THE MAIN PROGRAM OF THE FARANT
C
C*********************************************************************
C
C  ASSIGN COMMON DATA BLOCK TO :
C  IFLAG -- INDICATES THE SUCCESS OF AN OPERATION BY HAVING A VALUE
C           ZERO (INTEGER)
C  ZO -- CHARACTERISTIC IMPEDENCE (REAL)
C  F -- FREQUENCY (REAL)
C  ICOU -- INDICATES THE SIZE OF DB (INTEGER)
C  DB -- DATA BASE FOR STORAGE OF DATA TO BE PRINTED OR PLOTTED (REAL)
C
      DOUBLE PRECISION ZO,DB,PI,F
      CHARACTER*8 HMS,DMY*9
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,F,ICOU,DB(101,18)/B1/PI
      PI = 3.14159265358979323 8D0
      CALL FARSTART
      CALL DATE(DMY)
      CALL TIME(HMS)
      WRITE(6,10) HMS,DMY
   10 FORMAT('  TIME: ',A8,49X,'DATE: ',A9)
      STOP 'SUCCESSFUL EXIT                         FARANT VERSION 2.0'
      END
C************************************************************************
C
C  FUNCTION: CKTANALYSIS
C
C  INPUT: X,OPT
C
C  OUTPUT: FVAL, OPT
C
C  SUBROUTINES CALLED: SPECIFIED BY USERS
C
C  DESCRIPTION: CKTANALYSIS IS A SUBROUTINE WHICH IS WRITTEN BY THE
C               USER.  ITS PURPOSE IS TO CARRY OUT CIRCUIT ANALYSIS
C               AND EVALUATE THE OBJECTIVE FUNCTION.
C  FVAL -- THE VALUE OF THE OBJECTIVE FUNCTION (REAL)
C  X -- THE PARAMETERS TO BE OPTIMIZED; ITS MAXIMUN NUMBER IS 24 BUT IT
C       CAN BE MODIFIED (REAL)
C  OPT -- A FLAG USED FOR INDICATING WHETHER FVAL IS NEEDED WHEN
C         OPT = 1, FVAL IS NEEDED; WHEN OPT = 0, CARRY OUT NORMAL
C         CIRCUIT ANALYSIS. (INTEGER)
C
C************************************************************************
      SUBROUTINE CKTANALYSIS(X,FVAL,OPT)
```

```
      IMPLICIT REAL*8 (A-H,L,K,O-Z)
C
C  !!! DIMENSION OF X NEEDS TO BE CHANGED IF MORE THAN 24 PARAMETERS ARE
C       USED
C
      DIMENSION X(24)
C
C  !!! MORE ELEMENTS CAN BE ADDED HERE
C
      DIMENSION A(4,4),B(4,4),C(4,4),D(4,4),E(4,4),F(4,4),G(4,4),H(4,4)
      DIMENSION Q(2,3,12),P(3,4),TR(20),CIN(15),OUT(15),ER(4),X1(70)
      DIMENSION Y1(4,70)
      NAMELIST /INNET/ CIN /OUTNET/ OUT /NE673A/ TR
      CHARACTER ANS,CH(12)*18,VAS*21/'X AXIS DB'/,HAS*21/'Y AXIS FREQ'/
      CHARACTER TITL*76/'  1 - S11, 2 - S12, 3 - S21, 4 -S22'/
      CHARACTER CHA(4)/'1','2','3','4'/,NOF*20
      INTEGER OPT
C
C  !!! DATA BASE CAN BE INCREASED
C
      COMMON IFLAG,ZO,FREQ,ICOU,DB(101,18)/B1/PI
      ICOU = 0
      IFLAG = 0
C
C  !!! FARANT'S REF ZO IS ASSIGNED ONLY HERE
C
      ZO = 50.D0
C
C
+---------+---------+---------+---------+---------+---------+---------+
C    USER'S PROGRAM BEGIN IN THE FOLLOWING LINES
C
      IF (IR .EQ. 0) THEN
      FREQ1 = 21.6D0
      F2 = 23.6D0
      NF = 3
      DF = (F2-FREQ1)/(FLOAT(NF)-1.D0)
      WRITE(6,30)
   30 FORMAT('1 PLEASE TYPE THE SQUARE OF S PARAMETERS'/'  THE S ',
     1'PARAMETERS SHOULD BE ENTERED ON THE SAME LINE AS FOLLOWS:')
      DO 40 I = 1,NF
      FT = DF*(I-1)+ FREQ1
   32 WRITE(6,35) FT,I,I,I,I
   35 FORMAT('  AT ',G14.7,' GHz'/7X,'P(',I1,',1),',10X,'P(',I1,',2),',
     110X,'P(',I1,',3),',10X,'P(',I1,',4),')
      READ(5,*) (P(I,J),J = 1,4)
      WRITE(6,50) (I,P(I,J),J = 1,4)
   50 FORMAT('       P(',I1,',1) = ',G14.7,'     P(',I1,',2) = ',G14.7/
     1'       P(',I1,',3) = ',G14.7,'     P(',I1,',4) = ',G14.7/'  TYPE'
     1,' "Y" (IF DATA IS CORRECT) OR "N" (TO CHANGE DATA)'/' > ',$)
```

```
      READ(5,'(A)') ANS
      IF (ANS .NE. 'Y') GO TO 32
   40 CONTINUE
      TYPE *,' ENTER NAME OF THE FILE CONTAINING COMPONENT VALUES: '
      READ(5,'(A20)') NOF
      OPEN (UNIT=7,STATUS='OLD',FILE=NOF)
      READ(7,NML=INNET)
      READ(7,NML=OUTNET)
      READ(7,NML=NE673A)
      CLOSE(UNIT=7,DISP='KEEP')
      END IF
      IF (OPT .EQ. 0) THEN
        DO I = 1,NF
        FREQ = FREQ1 + (FLOAT(I) - 1.D0)*DF
        CALL INPUT(A,CIN)
        CALL PACK(A,Q,1,I)
        CALL OUTPUT(A,OUT)
        CALL PACK(A,Q,2,I)
        END DO
      END IF
      FVAL = 0.D0
      ICOU = 0
      TR(1) = X(5)*X(5)
      TR(3) = X(1)*X(1)
      TR(11) = X(2)*X(2)
      TR(6) = X(3)*X(3)
      TR(13) = X(4)*X(4)
      TR(10) = X(6)*X(6)
      DO 90 I = 1,NF
      FREQ = FREQ1 + (FLOAT(I) - 1.D0)*DF
      CALL UNPACK(G,Q,1,I)
      CALL FET(B,TR,0)
      CALL RLC(A,'P',0.D0,CIN(15),0.D0,'P',0.D0)
      CALL SER(A,B)
      CALL CAS(G,A)
      CALL UNPACK(A,Q,2,I)
      CALL CAS(G,A)
      CALL MTRANS(G,4,0)
      ER(1) = (G(1,1)**2+G(1,2)**2-P(I,1))**2
      ER(2) = (G(1,3)**2+G(1,4)**2-P(I,2))**2
      ER(3) = (G(2,1)**2+G(2,2)**2-P(I,3))**2
      ER(4) = (G(2,3)**2+G(2,4)**2-P(I,4))**2
      FVAL = FVAL + 10.D0*ER(1) + ER(2) + ER(3) + 10.D0*ER(4)
   90 CALL SAVECKT(G,4,0,-1.D0)
      IF (OPT .EQ. 1) RETURN
      WRITE(6,'(1H1)')
   95 DO 100 I = 1,ICOU
      X1(I) = DB(I,1)
      Y1(1,I) = 10.D0*LOG10(DB(I,2)**2+DB(I,3)**2)
      Y1(2,I) = 10.D0*LOG10(DB(I,4)**2+DB(I,5)**2)
```

```
      Y1(3,I) = 10.D0*LOG10(DB(I,6)**2+DB(I,7)**2)-30.D0
  100 Y1(4,I) = 10.D0*LOG10(DB(I,8)**2+DB(I,9)**2)
      CALL PLOT(X1,Y1,CHA,FREQ1,F2,-30.D0,0.D0,0,4,ICOU,VAS,HAS,TITL)
      IF (IR .EQ. 1) THEN
        WRITE(6,101)
  101 FORMAT('  COMPONENT VALUES OF THE INPUT CIRCUIT:'/T30,'ZO',15X,
     1'LENGTH'/1X,60('-'))
        WRITE(6,102) CIN
  102 FORMAT(' INPUT TRANSFORMER',T25,G14.7,5X,G14.7/' TUNING L',T25,
     1G14.7,5X,G14.7/' GATE LEAD',T25,G14.7,5X,G14.7/' GATE BIAS',T25,
     1G14.7,5X,G14.7/' GATE CAP',T25,G14.7,5X,G14.7//' DISCONT. CAP = ',
     1G14.7,10X,'END CAP = ',G14.7/' BIAS RESISTOR = ',G14.7,10X,'BIAS'
     1,' CAP = ',G14.7/' SOURCE INDUCTOR = ',G14.7)
        WRITE(6,103)
  103 FORMAT(/'  COMPONENT VALUES OF THE OUTPUT CIRCUIT:'/T30,'ZO',15X,
     1'LENGTH'/1X,60('-'))
        WRITE(6,104) (OUT(II),II=1,6),(OUT(II),II=10,13),(OUT(II),II=7,
     1  9),OUT(14),OUT(15)
  104 FORMAT(' DRAIN LEAD',T25,G14.7,5X,G14.7/' DRAIN BIAS',T25,G14.7,
     15X,G14.7/' DRAIN CAP',T25,G14.7,5X,G14.7/' T2',T25,G14.7,5X,G14.7
     1/' OUT LINE',T25,G14.7,5X,G14.7//' END CAP = ',G14.7,10X,'DISCON.'
     1,' CAP = ',G14.7/' DISCON. CAP = ',G14.7,10X,'BIAS BYPASS CAP',
     1' = ',G14.7/' BIAS BYPASS RESISTOR = ',G14.7)
      END IF
      CALL PRT(4,-4)
      WRITE(6,105) TR(1),TR(3),TR(11),TR(6),TR(13),TR(10)
  105 FORMAT(2X,'TR(1) = ',G14.7,' TR(3) = ',G14.7,' TR(11) = ',G14.7/
     1'  TR(6) = ',G14.7,' TR(13) = ',G14.7,' TR(10) = ',G14.7)
      IR = 1
      RETURN
      END
```

Fig. 10 Program listing of [SW.FARANT2]FITNE.FOR.

```
X     0|                                                                Y
  -15004
A -30001                                                                A
X -45001                            4                                 X X
I -60001                                                                I
S -75001                                                                S
  -90001                            1
D-105001                                                                F
B-120001                                                                R
 -135001                                                                E
 -150001                                                                Q
 -165001
 -180001
 -195001                            2
 -210001                                                              2
 -225001
 -240001                            3
 -255001                                                              3
 -270002
 -285001
 -300003---------+---------+---------+---------+---------+---------+---------+
E -3  21600   21886    22171    22457    22743    23029    23314    23600
  1 - S11, 2 - S12, 3 - S21, 4 -S22                                    E -3
 TYPE ANY CHARACTER TO CONTINUE. >
```

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 21.600 | 0.6775 | 160.5 | 0.0455 | 95.7 | 0.9944 | 95.6 | 0.9073 | 158.5 | 0.00 |
| 22.600 | 0.3375 | 124.9 | 0.1020 | 40.8 | 2.0365 | 40.8 | 0.6444 | 138.3 | 0.00 |
| 23.600 | 0.6010 | -133.2 | 0.0959 | -57.8 | 1.7537 | -57.8 | 0.5840 | -161.7 | 0.00 |

```
TYPE ANY CHARACTER TO CONTINUE >
TR(1) =    7.300804    TR(3) =   0.1000000E-03 TR(11) =   0.2933306
TR(6) =  0.5760000E-03 TR(13) =   0.3064730     TR(10) =    1.638400

INITIAL VALUES OF VARIABLES ARE:

  0.1000000E-01    0.5416000        0.2400000E-01    0.5536000

   2.702000         1.280000

INITIAL FUNCTION VALUE =    12.37276

SENSITIVITIES:
-0.20402        -141.59         0.91065        -426.78         49.667
  13.835


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


STEP #    1 [X]:
 0.11518E-01   0.56106        0.21176E-01   0.61098         2.7006
  1.2792
                                                    FVAL =    42.60838

STEP #    2 [X]:



STEP #   83 [X]:
-0.44502E-01   0.53734        0.55897E-01   0.56312         2.6484
  1.0853
                                                    FVAL =    7.269432
SENSITIVITY:
 0.22240E-05  -0.51820E-04   0.10471E-05  -0.79578E-04  -0.12386E-04
-0.17515E-05

STEP #   84 [X]:
-0.44502E-01   0.53734        0.55897E-01   0.56312         2.6484




  1.0853
                                                    FVAL =    7.269432
```

```
X     0|                                                                       Y
  -15004
A -3000|                                                                     4 A
X -4500|                                                                     1 X
I -6000|                                                                       I
S -7500|                                  4                                    S
  -9000|                                  1
D-10500|                                                                       F
B-12000|                                                                       R
 -13500|                                                                       E
 -15000|                                                                       Q
 -16500|
 -18000|                                  2
 -19500|
 -21000|                                                                     2
 -22500|                                  3
 -24000|
 -255002                                                                     3
 -27000|
 -285003
 -30000+---------+---------+---------+---------+---------+---------+---------+
E -3  21600   21886    22171    22457    22743    23029    23314    23600
   1 - S11, 2 - S12, 3 - S21, 4 -S22                                    E -3
```

TYPE ANY CHARACTER TO CONTINUE. >

COMPONENT VALUES OF THE INPUT CIRCUIT:

| | ZO | LENGTH |
|---|---|---|
| INPUT TRANSFORMER | 26.00000 | 0.1260000 |
| TUNING L | 50.00000 | 0.1450000 |
| GATE LEAD | 100.0000 | 0.1000000E-01 |
| GATE BIAS | 90.00000 | 0.1250000 |
| GATE CAP | 39.00000 | 0.7500000E-01 |

DISCONT. CAP = 0.0000000E+00 END CAP = 0.4000000E-01
BIAS RESISTOR = 50.00000 BIAS CAP = 0.5000000
SOURCE INDUCTOR = 0.1000000E-02

COMPONENT VALUES OF THE OUTPUT CIRCUIT:

| | ZO | LENGTH |
|---|---|---|
| DRAIN LEAD | 100.0000 | 0.1000000E-01 |
| DRAIN BIAS | 90.00000 | 0.1250000 |
| DRAIN CAP | 39.00000 | 0.4000000E-01 |
| T2 | 15.00000 | 0.1260000 |
| OUT LINE | 50.00000 | 0.0000000E+00 |

END CAP = 0.4000000E-01 DISCON. CAP = 0.3000000E-01
DISCON. CAP = 0.3000000E-01 BIAS BYPASS CAP = 0.5000000
BIAS BYPASS RESISTOR = 50.00000

[S] PARAMETERS IN MAGNITUDE AND PHASE

| | 11 | | 12 | | 21 | | 22 | | K |
|---|---|---|---|---|---|---|---|---|---|
| FREQ | MAG | ANG | MAG | ANG | MAG | ANG | MAG | ANG | FACT |
| 21.600 | 0.7209 | 160.1 | 0.0542 | 98.3 | 1.1384 | 92.2 | 0.8742 | 156.6 | 0.00 |
| 22.600 | 0.3611 | 106.3 | 0.1289 | 34.2 | 2.4704 | 27.4 | 0.4469 | 135.8 | 0.00 |
| 23.600 | 0.6029 | -135.2 | 0.0887 | -64.0 | 1.5565 | -71.6 | 0.7496 | -163.5 | 0.00 |

TYPE ANY CHARACTER TO CONTINUE >
TR(1) = 7.014131 TR(3) = 0.1980460E-02 TR(11) = 0.2887337

TR(6) = 0.3124524E-02 TR(13) = 0.3171046 TR(10) = 1.177848
STATISTICS FOR THIS JOB:
ELAPSED TIME = 186.2891 SEC.
CPU TIME = 76810 MS BUFFER I/O COUNTED = 0
DIRECT I/O COUNTED = 0 PAGE FAULTS COUNTED= 0
TIME: 15:58:38 DATE: 16-AUG-83

Fig. 11 Results of the optimization program using FARANT 2.0

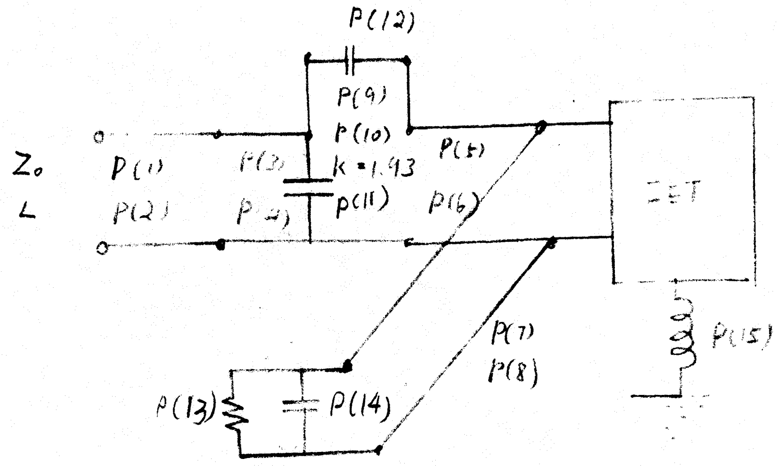
```
$INNET
CIN=26,0.126,50,.145,100,.01,90,.125,39,.075,0,.04,50,0.5,.001
$END
$OUTNET
OUT=100,.01,90,.125,39,.04,.04,.03,.03,15,.126,50,0,.5,50
$END
$NE673A
TR=,0.42,,0.2,1.55,,369,49,0.3,,,0.26,,0.26,0,13.9,207,805,.079,0
$END
```

Fig. 12 Listing of FIT2AUG83.

Several points deserve the attention of the user:

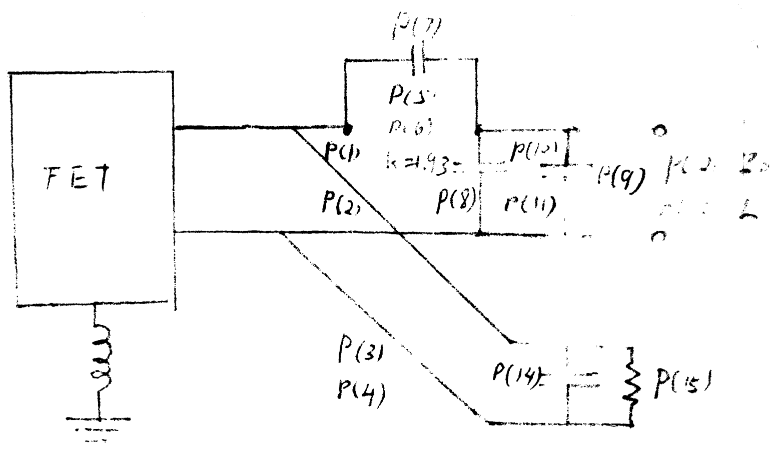
1. The program is divided into three portions -- a) reading data and finding out two-port descriptions for input and output networks, b) the actual beginning of circuit analysis, putting three networks -- input and output circuits and FET model -- together and calculating of FVAL; and c) printing tables and graphs.

2. The portion a) of the program is only executed once by using the flag, IR. It also makes use of the **VAX's** special feature -- the saving of local variables of a subroutine after leaving the subroutine. Therefore, those parameters read will stay and can be used on the second call of the subroutine CKTANALYSIS.




| K | P(K) |
|---|---|
| 1 | 26 Ohms |
| 2 | 0.126 in |
| 3 | 50 Ohms |
| 4 | 0.145 in |
| 5 | 100 Ohms |
| 6 | 0.01 in |
| 7 | 90 Ohms |
| 8 | 0.125 in |
| 9 | 39 Ohms |
| 10 | 0.075 in |
| 11 | 0.0 pF |
| 12 | 0.04 pF |
| 13 | 50 Ohms |
| 14 | 0.5 pF |
| 15 | 1. µH |

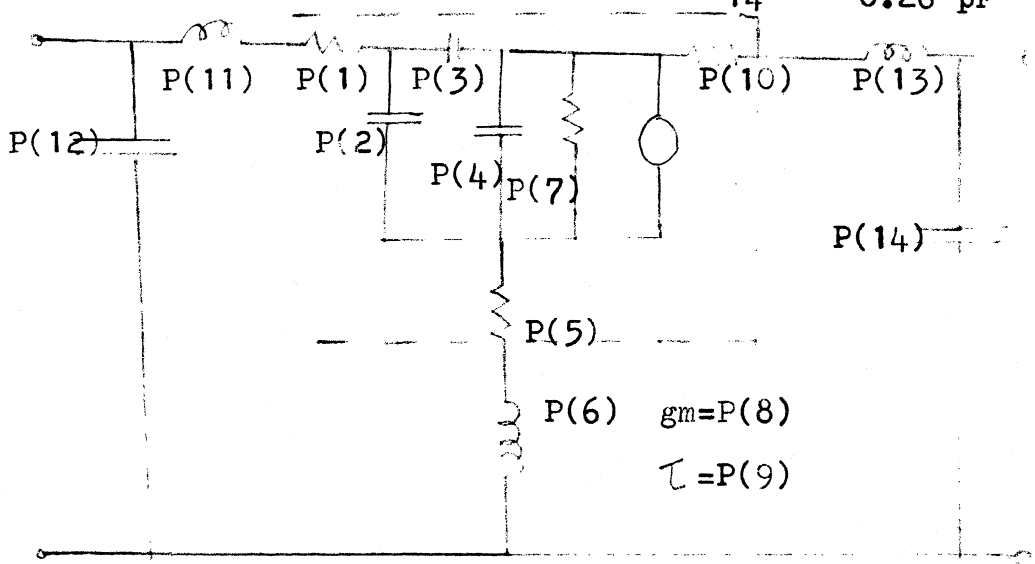
Fig. 13 The circuit diagram of the input network.




| K | P(K) |
|---|---|
| 1 | 100 Ohms |
| 2 | 0.01 in |
| 3 | 90 Ohms |
| 4 | 0.125 in |
| 5 | 39 Ohms |
| 6 | 0.04 in |
| 7 | 0.04 pF |
| 8 | 0.03 pF |
| 9 | 0.03 pF |
| 10 | 15 Ohms |
| 11 | 0.126 in |
| 12 | 50 Ohms |
| 13 | 0.0 in |
| 14 | 0.5 pF |
| 15 | 50 Ohms |

Fig. 14 The circuit diagram of the output network.

| K | P(K) |
|---|---|
| 1 | * |
| 2 | 0.42 pF |
| 3 | * |
| 4 | 0.2 pF |
| 5 | 1.55 Ohms |
| 6 | * |
| 7 | 369 Ohms |
| 8 | 49 mMhos |
| 9 | 0.3 psec |
| 10 | * |
| 11 | * |
| 12 | 0.26 pF |
| 13 | * |
| 14 | 0.26 pF |




* Elements to be optimized

Fig. 15 The Fet model used.

3. The OPT flag is used for obtaining plotting and printing of parameters are used in portion c) of the program.

4. DO statement in **FORTRAN** is used instead of FOR and NEXT statements.

5. **FARANT 1.0** took about 70 seconds to execute the program while **FARANT 2.0** takes about 60 seconds.

6. Both versions take about 85 steps to minimize the objective function, and the objective function is minimized to a value about 7.27.

## 4.0 REFERENCES


1. Dan L. Fenstermacher, "A Computer-aided Analysis Routine Including Optimization for Microwave Circuits and Their Noise," EDIR No. 217, 1981

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

3. "VAX-11 Fortran User's Guide," 1982, PP. 6-1 to 7-20.

4. "VAX/VMS I/O User's Guide (Volume 1)," 1982, pp. 9-1 to 9-43.

5. "VAX/VMS System Service Reference Manual," 1982, pp. 6-1 to 5-14, pp. 167-172, pp. 124-132.

6. "VAX-11 Fortran User's Guide," 1982, pp. 1-19 to 1-22.

7. "VAX-11 Fortran Language Reference Manual," 1982, pp. C-1 to C-34

8. "VAX/VMS Guide to Using Command Procedures," 1982, pp.3-1 to 3-20, pp. 4-1 to 4-10, pp. 6-1 to 6-10, pp. 7-1 to 7-10, pp. 8-1 to 8-7.

9. "VAX-11 Fortran Language Reference Manual," 1982, pp. 9-1 to 9-16.

10. "VAX-11 Fortran Language Reference Manual," 1982, pp. 7-18 to 7-22.

## ACKNOWLEDGMENTS

The author would like to thank Dr. Sandar Weinreb, Dr. John Granlund and Dr. L. R. D'Addario of Electronics Division in Charlottesville for their guidance, direction and encouragement in translating and changing **FARANT.** He also wishes to express his appreciation to Mr. Fred Schwab for providing frequent consultation in using the **VAX.**