NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VIRGINIA

ELECTRONICS DIVISION INTERNAL REPORT NO. 250

FARANT ON THE HP9816 COMPUTER

JOHN GRANLUND

JULY 1984

NUMBER OF COPIES: 150

FARANT ON THE HP9816 COMPUTER

John Granlund

TABLE OF CONTENTS

FIGURES

Note: Comments on HP9816 BASIC are scattered throughout this report.

FARANT ON THE HP9816 COMPUTER

John Granlund

It is not intended that these pages contain a complete description of FARANT, nor should they constitute an instruction manual for HP9816 BASIC. Instead, only changes in FARANT since the publication of Fenstermacher's EDIR No. 217*, part of which describes FARANT on the HP9845 computer, will be covered. Certain features of HP9816 BASIC will be presented as they pertain to the discussion. Some of the reasons for the changes are as follows:

- HP9816 BASIC is different from HP9845 BASIC.
- New requirements for FARANT have emerged and have been implemented.
- A library of FET SUBprograms has been started; libraries for other disciplines should be considered as they are needed.

1.0. The HP9816 Computer -- Getting Started

The HP9816 has ten times the memory of the HP9845, and it seems to operate about three times faster. A big difference is that HP9845 BASIC is available to its computer from a ROM at power-on, whereas at power-on, the HP9816 only has instructions to test its memory and then to search the connected disc drives for an operating system: HP9816 BASIC is booted from a disc. This makes it possible for Hewlett-Packard to extend or correct HP9816 BASIC simply by issuing new operating system discs.

---

*Dan L. Fenstermacher, "A Computer-Aided Analysis Routine Including Optimization for Microwave Circuits and Their Noise," NRAO Electronics Division Internal Report No. 217, June 1981.

HP9816 BASIC is actually stored on 1-1/2 discs. The computer finds and boots the contents of the BASIC System disc at power-on, and this provides it with the capabilities of BASIC 2.0, which do not include matrix algebra, typing-aid keys, and a number of other features collectively called Extended BASIC 2.0†. These features are contained in the binary file AP2_0, which must be LOADed next or not at all: They are needed for FARANT. The disc drive in which the computer finds the BASIC System disc becomes the default drive. The other drive can, of course, be used, but only after supplying its "mass storage unit specifier" -- MSUS* -- to the computer. It is recommended that, if possible, the default drive be used exclusively throughout each session with the computer. However, to equalize drive wear, the default drive should be switched from one session to the next.

Programs can be SAVEd in ASCII files on the disc and retrieved by GET, with an option to name their first line number after automatic RENumbering. Programs can also be STOREd in PROGram files and retrieved by LOAD, without the first line number option, however. In fact, if FARANT is LOADed and then FET_LIB, the FET library, is LOADed, all of FARANT is SCRATCHed before the second LOAD! The LOADSUBprogram command offers a way around this difficulty, because only FARANT contains a mainline program that must be LOADed. In spite of the flexibility of SAVE and GET, STORE and LOAD -- or LOADSUB -- are recommended

---

†The updated Extended Basic 2.1 is now available from the binary file AP2_1.

*MSUS is the first line of printout resulting from a CATalog command.

because they are much faster. The command

```
LOADSUB Optimize FROM "FARANT"
```

would install just SUBprogram Optimize, renumbered in steps of 5, to follow 5 steps after the last step that was in program memory.

```
LOADSUB ALL FROM "FET_LIB"
```

would install all of the SUBprograms from the FET_LIB file immediately after what had been the last step in program memory.

It should be noted that, in EDITing a program numbered in steps of 5, as many steps as are desired -- specifically, more than 4 -- can be inserted between any two lines: As many of the lines below the insertion point as is necessary are automatically RENumbered. A step size of 5 has been chosen for all of FARANT and is recommended for user's SUBprograms. Line numbers through 9995 are reserved for FARANT, line numbers from 10000 through 19995 are reserved for a library, and line numbers from 20000 through 32766 are for user's SUBprograms.

At this time, the last line of the last FARANT SUBprogram is numbered 8040. To deal with the close RENumbering provisions of the LOADSUB command, the line of comment

```
9995 !########################## LIBRARY IS NEXT ##########################
```

has been added to FARANT. HP9816 BASIC considers this line to belong to the last SUBprogram of FARANT. Since FET_LIB is numbered in steps of 5, the command LOADSUB ALL FROM "FET_LIB" brings in the library, numbered starting with line 10000. Line 19995 contains a corresponding comment so that when user's SUBprograms are brought in with the LOADSUB command, they will be numbered starting with line 20000.

For example, the following steps are appropriate after FET_LIB has been EDITed:

DELete FARANT, including its tacked-on line of comment.

DELete user's SUBprograms.

RENumber FET_LIB, starting at line 10000, in steps of 5.

Make sure that the line number of the last comment is 19995.

STORE "FET_LIB"

Attempting to STORE "FET_LIB" on the same disc from which it was LOADed evokes

ERROR 54 Duplicate file name

Using the RE-STORE command creates a second FET_LIB file on the disc. Only when this STORE operation has been successfully completed is the address of the first version of FET_LIB removed from the disc's directory, leaving an unused section of storage on the disc. To maintain a tightly-packed collection of files on the disc, one must have the files STOREd after FET_LIB also STOREd on another disc. Note that this may be a difficult position to reach, starting with the newly-EDITed version of FET_LIB in the computer's program memory. To re-establish tightly-packed file storage, the old version of FET_LIB and the following files on the disc are PURGEd, one at a time, the new FET_LIB is STOREd, and each of the following files is LOADed into program memory and STOREd.

The HP9845 STORE ALL and LOAD ALL commands are not available in HP9816 BASIC. This means that the user must load typing-aid key definitions separately, and that he will not be greeted by Fenstermacher's flashy start-up message for the HP9845. The following start-up procedure is recommended:

Turn on the HP9121 dual disc drive and HP2673A printer.

Select the default disc drive and install the BASIC System disc.

Turn on the HP9816 computer and allow BASIC 2.0 to be booted.

Install the FARANT disc in the default drive.

Type and EXECute the following commands:

LOAD BIN "AP2_0"

LOAD KEY "FARKEY" -- this step can be repeated whenever necessary.

LOAD "FARANT"

LOADSUB ALL FROM "FET_LIB"

LOADSUB ALL FROM "FARAFT" -- or from an appropriate version of FARAFT with a different file name.

2.0 Organization of FARANT for the HP9816

Figure 1 displays abbreviated listings of the FARANT, FET_LIB, and FARAFT files, and it names four files containing other versions of FARAFT. Fenstermacher's FARANT, described in EDIR No. 217, is contained in the files FARANT and FARAFT. It is intended that FARAFT -- or one of its versions -- contain all of a user's program and be STOREd and maintained by the user. Since SUBprograms Nread and Pread are arranged to contain the user's noise and signal data, they are properly part of the FARAFT file, rather than FARANT SUBprograms. Four SUBprograms containir data used in some of the FET_LIB SUBprograms also fall in this category. If they are not needed, the user can DELete these SUBprograms before installing the major portion of his work in SUB Cktanalysis.

2.1 Changes in the FARANT File

It will be noted from the abbreviated listings that INTEGER variables are now used as pass parameters for many of the SUBprograms. The intent is to speed up both the CALL operation and the internal execution of the SUBprogram itself. In HP9816 BASIC, a variable is a REAL -- a floating point number represented as in the IEEE 64-bit standard -- by default. Only if it is specified to be an INTEGER is it -- and its sign -- represented by only 16 bits. Furthermore, in COMmon, DIMension, and ALLOCATE statements; in SUBprogram statements; and

```
        FARANT mainline and SUBprograms:
370   SET TIMEDATE DATE(D$)+TIME(T$)
400   SUB Mtrans(X(*),INTEGER Pset)     !NEW 2-PORT PARAMS OF [X]    #########
1495  SUB Ntrans(X(*),INTEGER Nset)                                  !#########
2210  SUB Rlc(X(*),Type$,R,L,C,Place$,Tamb)                          !#########
2495  SUB Source(X(*),Control$,Source$,Gain,R1,R2,Delay)             !#########
2635  SUB Trline(X(*),Zg,Length,K)                                  !#########
2710  SUB Tf(X(*),Turns1,Turns2)                                    !#########
2750  SUB Lossyline(X(*),Zgo,Length,K,Cattn,Dattn,Fo,Tamb)          !#########
3040  SUB Flip(X(*))                    !SWAPS INPUT AND OUTPUT PORTS  #########
3220  SUB Flop(X(*))                  !SWAPS INPUT & COMMON TERMINALS  #########
3425  SUB Branch(X(*),Type$)                                        !#########
3620  SUB Nload(X(*),N1,N2,N3,N4,INTEGER Nset)                      !#########
3805  SUB Polar(X,Y)                                                !#########
3845  SUB Cas(X(*),A(*))                                            !#########
4100  SUB Lstrip(A(*),X(*))    !REMOVES A(*) FROM THE LEFT OF X(*)   #########
4400  SUB Rstrip(X(*),A(*))    !REMOVES A(*) FROM THE RIGHT OF X(*)  #########
4705  SUB Par(X(*),A(*))                                            !#########
4780  SUB Ser(X(*),A(*))                                            !#########
4855  SUB Saveckt(X(*),Kfact,INTEGER Pset,Nset)                     !#########
5130  SUB Nperformance(X(*),Rs,Xs,Rl,Xl,In,Gain,INTEGER Gtype)      !#########
5410  SUB Prt(INTEGER Pset,Nset)                                    !#########
6055  SUB Smith(Xmin,Xmax,Ymin,Ymax)                                !#########
6310  SUB Splot(INTEGER I,J)                                        !#########
6375  SUB Gammaz(U,V,R,X,INTEGER Option)                            !#########
6515  SUB Zio(X(*),Rs,Xs,Rl,Xl,Rin,Xin,Rout,Xout)                   !#########
6675  SUB Optimize(X(*),INTEGER Dtype)                              !#########
7425  SUB Vectprt(A$,X(*),B$,INTEGER N)   !FOR 75-COLUMN PRINTER    #########
7525  SUB Matprt(X(*))                     !FOR DEBUG:. PRINTS 12   #########
7660  SUB Clock(Date$,Time$)                                 !FORMATS: #########
7725  SUB Letter                                                    !#########

        FET_LIB SUBprograms:
10000 SUB Fet1(A(*),P(*))       !Loads A(6,4) with 2-port params.  #########
10220 SUB In1(A(*),P(*))                !Loads A(6,4) with input    #########
10325 SUB Out1(A(*),P(*))              'Loads A(6,4) with output    #########
10450 SUB Mplot(INTEGER Gmin,Gmax,Tmax,Opt) !Opt=0: In(50), 0-Tmax#########
10960 SUB Web(A(*),INTEGER Opt)      !Input: Opt=1, Output: Opt=2  #########

        FARAFT SUBprograms:
20000 SUB Farstart          !USER'S CONTROL OF FARANT BEGINS HERE #########
20050 SUB Cktanalysis(X(*),Fvalue,INTEGER Opt)                    !#########
20105 SUB Nread(X(*))                                             !#########
20290 SUB Pread(X(*))                                             !#########
20545 SUB Ne673a(P(*))              !Loads P(20) with FET params.   #########
20570 SUB Mgf1412a(P(*))            !Loads P(20) with FET params.   #########
20595 SUB K1(P(*))       !Loads P(15) with input network params.    #########
20620 SUB K2(P(*))      !Loads P(17) with output network params.    #########

        Other versions of FARAFT:
      SLICE1
      673A_FIT
      673A_MFIT
      ROSENBROCK
```

Figure 1. Abbreviated Listings

in INTEGER statements, all variables following the term INTEGER are taken to be INTEGERs until a variable name preceded by REAL is met, from which point on, all following variables are again taken to be REALs. To avoid the need to use both the terms INTEGER and REAL in the SUBprogram pass parameter lists, the order of naming the pass parameters has been changed in SUBprograms Nload, Saveckt, Nperformance, and Gammaz.

While on this subject, it should be noted that, with INTEGER K in effect, setting K=SQR(3) is perfectly acceptable in HP9816 BASIC: SQR(3) is rounded so that K=2. But with INTEGER K <u>not</u> in effect, setting or calculating K=3 and then CALLing Mtrans (X(*),K) stops the program with an error message. In this case K is certainly an integer, but, as required, it is <u>not</u> an INTEGER. CALLing Mtrans(X(*),3) works as expected.

Finally, it is not surprising that, with INTEGER K in effect, calculating the number of seconds in a year by

```
K=60*60*24*365
```

will result in an INTEGER overflow error, because the product is greater than 2^{15}. But it <u>is</u> surprising that this same error message appears with REAL K in effect! If the product were small enough, it would be calculated starting from the left and incorporating the next factor to the right, in this case using three steps, all in INTEGER arithmetic. Only then would the product be converted to a REAL and stored in K. But in calculating this product, INTEGER overflow occurs in the second step. HP9816 BASIC is performed in INTEGER arithmetic wherever possible, but it reverts to REAL arithmetic when one of the terms being combined is a REAL or when a division is called for. "60" and "32000", as they stand, are INTEGERs, but "60.", "120/2", and "33000" are REALs, the last because

it exceeds 2^{15}. Evidently, a simple "fix" for the previous calculation would be

```
K=60*60*24.*365
```

The FARANT mainline program has been extended from its original four steps to allow the user to set the HP9816 TIMEDATE clock once and for all after power-on, seven SUBprograms have been added to the FARANT file, and some changes have been made to SUB Optimize to make optimization easier and more flexible for the user. A discussion of these items follows.

The Mainline Program

Once its clock is set, the HP9816 computer is arranged to maintain the Julian date and time of day to well beyond the expected life of the computer, but this time is lost whenever the computer is turned off or power is lost. The two FET_LIB plot SUBprograms call for the date and time, and it is recommended that the user apply the date and time to his work. To allow the clock to be set, the mainline program presents the user with the computer's current date and time -- very likely 1 Mar 1900, a few minutes after midnight -- ready for EDITing, the first time after power-on that FARANT is RUN. Because of the method of presentation, the user must press ENTER <u>twice</u> after his EDITing. The first press will evoke

```
ERROR 120  Not allowed while prog running
```

but the second press will start the wheels turning again.

If, at some point in the execution of his program, the user finds that he has mis-set the TIMEDATE clock, he need not SCRATCH program memory and start all over again from from scratch. EXECuting SCRATCH Common, followed by RUN, will provide another opportunity to set the clock. If COMmon contains data

that the user wants to retain, he should STOP the program -- PAUSE is sufficient only if EXECuted while within the mainline program -- type and EXECute Fl=0 -- or FL=0 -- and then RUN.

TIMEDATE is the clock's reading, in seconds, updated every 10 milliseconds, from the year 4712 BC. It is also useful for timing the execution of a program: T1=TIMEDATE is installed as the first program statement and T2=TIMEDATE as the last. Then the run time, in seconds, is T2 - T1. To avoid exhibiting roundoff error, it may be desirable to present this result using a formatted PRINT statement or PROUND(T2-T1,-2), which would limit the result to two digits to the right of the decimal point.

SUB Flop(X(*))

Like SUB Flip, which interchanges input and output ports, this SUBprogram manipulates the 2-port description that it receives in X(6,4) to produce a 2-port description -- also in X(*) -- of the device that would result if the input and common terminals of the original device were interchanged. Both signal and noise descriptions are appropriately modified. Thus

Flop(common emitter) = common base

and even

Flip(Flop(Flip(common emitter))) = emitter follower.

SUB Lstrip(A(*),X(*)) and SUB Rstrip(X(*),A(*))

These SUBprograms undo the work of SUB Cas. SUB Lstrip "uncascades" the 2-port described by A(6,4) from the left -- input -- side of the 2-port described by X(6,4) and returns the result in X(*); SUB Rstrip "uncascades" A(*) from the right side of X(*). It is intended that these SUBprograms be used to determine the 2-port description of an amplifier or other device that had to be measured

through known input and output coupling networks. Either both A(*) and X(*) should contain valid noise descriptions, or both should be noiseless: The result of stripping a noisy A(*) from a noiseless X(*) is not physical.

SUB Vectprt(A$,X(*),B$,INTEGER N)

The need for this SUBprogram originates with the options that have been chosen for the HP2673A printer. This is an 80-column printer, but the 80-column print field is 8 inches wide, leaving no room at the left edge of an 8-1/2 x 11-inch sheet of paper for the punched holes that one would like to use to install the printout in a notebook. To make room, the printer itself has been programmed to start its lines in column 6, so the printer is now a 75-column printer. With only this change, the printer, handed an 80-column line to PRINT, would simply discard the last 5 characters. To avoid this potential loss, the "line wrap around" option has also been chosen for the printer: Unless the characters in columns 76 through 80 are blanks, they are PRINTed separately on the next line. Now regardless of <u>printer</u> programming, the HP9816 computer responds to the program statement "PRINT X(*);" by organizing the present contents of the print buffer, followed by the elements of X(*), into compact 80-column strings, all but the last of which carries the carriage-return/line-feed command at its end. The ungainly result of asking the modified HP2673A printer to "PRINT X(*);" is illustrated below:

```
 * * * * | * * * * | * * * * | * * * * | * * * * | * * * * | * * * * | * *

-4.9903579E-20 -3.8180517E-16  1.3903302E+6 -4.6301186E+12  361001.57  3068
60.94
 .0039491177 -6.4045427  8.5739267E+13  9.9969658E-17  1.327081E-13 -14.806
826
 2.6267088E+16  1.8919698E-6 -5.4833714E-9  4.1980379E-10 -1332.6056
 1.0320025E-8 -.0015767144
```

The result of CALLing SUB Vectprt to PRINT the same X(*) is shown next:

```
 * * * * | * * * * | * * * * | * * * * | * * * * | * * * * | * * * * | * *

Start message  -4.9903579E-20  -3.8180517E-16  1.3903302E+6  -4.6301186E+12
361001.57  306860.94  .0039491177  -6.4045427  8.5739267E+13  9.9969658E-17
1.327081E-13  -14.806826  2.6267088E+16  1.8919698E-6  -5.4833714E-9
4.1980379E-10  -1332.6056  1.0320025E-8  -.0015767144  End message
```

SUB Vectprt PRINTs a vector -- a one-dimensional array -- with leading comment A$ and trailing comment B$, either of which can be omitted by setting A$ = "" and/or B$ = "". Neither A$ nor B$ nor any of the vector elements is split and PRINTed on two lines. Each line of PRINT is confined in length to 75 characters. Before being PRINTed, the elements of X(*) are rounded to contain only the number of significant figures specified by INTEGER N.

SUB Vectprt can, of course, also be used when the PRINTER IS the CRT, but since the print field is then 80 columns wide, "PRINT X(*);" does an equally neat job that fills the print field.

Consider, finally, the LISTing of a program with the modified HP2673A printer. HP9816 BASIC allows a single program line -- a single line number and associated statements -- to contain as many as 160 characters; it can be two 80-column lines long. But the modified printer would produce an ungainly mess in PRINTing such a line. To ensure a neat program LISTing, each line must be less than 76 characters long, exclusive of the start underline and stop underline commands, which the printer deals with by underlining.

SUB Matprt(X(*))

This SUBprogram PRINTS 12 significant figures of X(4) or X(N,4), any positive INTEGER N, at one matrix row per line. Its primary usefulness is in making a hard copy of a 2-port description X(*) reached at some point in a FARANT SUBprogram, for debugging purposes. For example, what is contained in Aa(4,4)

after step 4245 of SUB Lstrip? TRACE PAUSE 4250 is EXECuted, the program is RUN, and at the PAUSE, the following are typed and EXECuted

```
CALL Matprt(Aa(*))
TRACE OFF -- if one printout is enough; leaving TRACE PAUSE in effect
             slows program execution, even if the PAUSE step is never
             reached again.
```

and the program is CONTinued. Including "CALL" in the CALL command is only required by HP9816 BASIC if the command is executed from the keyboard or if it is not the first item in a program statement following the line number, as for example, in

```
20375    IF A(5,1)<>3 THEN CALL Mtrans(A(*),3)
```

SUB Clock(Date$,Time$)

This SUBprogram delivers the date and time from the TIMEDATE clock in separate, compact strings suitable for printout. The formats are

```
Date$ = "2/24/84"    and    Time$ = "0927"
```

Although Time$ is always a 4-character string, Date$ may contain 6, 7 or 8 characters.

SUB Letter

HP9816 BASIC offers no convenient way to letter a graph, as HP9845 BASIC does. This SUBprogram is an attempt to fill that need. The characters to be LABELed are produced in the default CSIZE 5,.6 -- about the smallest that is really legible after DUMP GRAPHICS. A flashing cursor is provided by using PEN 0, which changes whatever it writes on from dark to light and vice versa. Note that no "cursor tracks" are thus left on the graph, provided only that the cursor is LABELed an even number of times with PEN 0 in each position it

occupies. The interrupts required for this LABELing and parity check make it quite possible that the character called for by the user's next keystroke is missed, or worse, is only partially recorded, at which point the SUBprogram must search again for a valid recorded keystroke. As a result of these difficulties, SUB Letter's response to one keystroke may take noticeably more time than its response to another. The user's best defense is to type slowly, watching his graph to make sure that each keystroke receives the desired response.

Detailed user instructions for SUB Letter are incorporated into the SUBprogram LISTing and also appear below:

```
        T   Y
D   F   G   H   J
      V   B
   [SPACE BAR]
```

> You "fly" the cursor, with CTRL pressed, about G. F and H provide "small" -- 1/5 of a character space -- motions left and right, respectively. Full motions are provided by D and J. Similarly, T and B provide "small" up and down motions, and Y and V provide the full motions. When the cursor is centered under the most recently LABELed character, the SPACE BAR erases that character. G exits this lettering SUBprogram. When the cursor is appropriately located, release CTRL and use the typewriter slowly.

SUB Optimize(X(*),INTEGER Dtype)

This SUBprogram has been modified to PRINT partial results on the CRT, thus saving many pages of paper during a long RUN. Only initial guesses and final results are PRINTed by the printer selected by the user in SUB Farstart or SUB Cktanalysis. But if the user decides that he wants a hard copy of the next several Steps of optimization, he can PAUSE the program, type and EXECute PRINTER IS 701, and CONTinue. EXECuting PRINTER IS 1 will return the partial results to the CRT again. Alternately, the user can use the appropriate typing-aid

key without even bothering to PAUSE the program, if these keys have not been SCRATCHed and if he can remember which is the appropriate key: Typing-aid key labels are not displayed while the program is running.

Even the pass parameters with which SUB Optimize is CALLed by SUB Farstart have been changed. N, the number of variables to be adjusted in the optimization process, is obtained by SUB Optimize from the vector X(*) containing their initial estimates, by using the SIZE function. N has been replaced by INTEGER Dtype, following X(*), in the pass parameter list. Dtype advises SUB Optimize of the type of derivatives to use in evaluating the gradient of Fvalue, the objective function. In the course of optimization, SUB Optimize will CALL

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

with Opt = Dtype; as in previous versions of FARANT, the final CALL is made with Fvalue = Opt = 0 -- no response expected.

Dtype = 1 means numerical derivatives, which SUB Optimize obtains by incrementing each of the variables in X(*), one at a time. SUB Cktanalysis is then expected to return only Fvalue, as quickly as possible. This was the only gradient evaluation available in earlier versions of FARANT. But SUB Optimize will produce much more accurate results if the gradient is evaluated analytically. RUN time -- but almost certainly not analysis and programming time -- may also be substantially reduced if analytic derivatives are used to evaluate the gradient. For certain problems that will be repeatedly worked with different parameters, it may well be appropriate to use analytic derivatives. Dtype = 3 means analytic derivatives. After calling SUB Cktanalysis with Opt = Dtype = 3, SUB Optimize will interpret the returned X(*) as the gradient

$$G(*) = \frac{\partial Fvalue}{\partial X(1)} , \frac{\partial Fvalue}{\partial X(2)} , \ldots \frac{\partial Fvalue}{\partial X(N)} ,$$

and omit its own numerical calculation of derivatives. It will continue to

take the returned Fvalue to be the objective function at the point X(*) that it passed to SUB Cktanalysis.

In many cases, the user may find, by watching the partial results on the CRT, that the solution to his problem being reached by SUB Optimize is not practical. He would like to terminate optimization as quickly as possible, but only after obtaining the final printout and graphics that he has arranged for SUB Cktanalysis to produce when it receives the final CALL from SUB Optimize with Opt = 0. In other cases, the user may wish to investigate the effects of Eps, the SUB Optimize "quit" parameter, on RUN time and final accuracy. The default values of Eps are as tabulated next.

Dtype	Derivatives	Default Eps
1	Numerical	1.E-15
3	Analytic	1.E-30

Both of the above user objectives have been arranged for by making SUB Optimize responsive to the flag Opt as it is returned by SUB Cktanalysis: If INTEGER Opt is negative after a CALL of SUB Cktanalysis, SUB Optimize resets

$$\text{Eps} = 10^{\text{Opt}} .$$

To terminate optimization, the user should determine the line number of his version of SUB Cktanalysis where this SUBprogram is EXITed with the values requested by SUB Optimize stored in the pass parameters, and the line number of the last step of SUB Cktanalysis. For example,

```
:
:
20310   Fvalue=Fvalue/(4*N)
20315   IF Opt=1 THEN SUBEXIT
:
:
20570 SUBEND
```

With these line numbers, he should PAUSE the program, type and EXECute TRACE PAUSE 20315, and CONTinue. At the PAUSE, he should type and EXECute

```
TRACE OFF
Opt=-1
CONT 20570
```

The reset value Eps = 10^{-1} will almost certainly stop optimization at the next Step. If several Steps of optimization have already been completed, an even smaller value of Eps should suffice to terminate optimization.

The user is warned <u>not</u> to include in SUB Cktanalysis the program statement

```
Opt=-10
```

if that statement will be encountered more than once in the course of optimization, and for the same reason, he should <u>not</u> compute

$$|A + jB|^2 = A\hat{}2+B\hat{}2$$

but rather he should use

$$|A + jB|^2 = A*A+B*B$$

The reason for these warnings is that HP9816 BASIC responds to all exponentiations using

$$X\hat{}N = EXP(N*LOG(X)),$$

regardless of how small an integer N may be. Needless to say, the HP9816 BASIC computation of the natural log and the exponential function take lots of unnecessary time when N is a small integer. HP9845 BASIC uses a much more efficient

exponentiation algorithm when N is a suitably small integer. For example, for N = 37, it fills the vectors

Binary representation of N = 1 , 0 , 0 , 1 , 0 , 1

$X^{2^n} = X^{32}$, X^{16} , X^{8} , X^{4} , X^{2} , X ,

the second by starting from the right and squaring each vector element to obtain the next element to the left. Finally, those vector elements corresponding to 1's in the binary representation of N are multiplied together. In this way, X^{37} takes a mere 7 multiplies; X^2, of course, takes only one. If N is a negative integer, HP9845 BASIC calculates $X^{|N|}$ and then calculates its reciprocal.

In a preliminary comparison of RUN times, an optimization program was RUN on the HP9845 computer, and its translation was RUN on the HP9816 computer. The HP9816 computer RUN took 2/3 of the time required by the HP9845 computer. But the program used exponentiation to determine the squared magnitudes of certain complex numbers. With X^2 replaced by X*X in the HP9816 program, its RUN time was only 1/3 of the time required by the HP9845 computer.

X^N should be avoided in HP9816 BASIC!

2.2 The FET Library

At this time, the FET_LIB file contains three circuit SUBprograms and two plot SUBprograms. Each circuit SUBprogram is CALLed with a vector of circuit-element values that can be filled by CALLing one of the associated circuit-element value SUBprograms in the FARAFT file. These value SUBprograms contain nothing but a READ statement and two DATA statements; they provide a convenient way to STORE the required values with the program, without requiring the user to be repeatedly concerned with submitting the correct values to the FET_LIB circuit SUBprograms.

SUB Fet1(A(*),P(*))

Given the frequency F from COMmon, and having been passed the required values in P(20), this SUBprogram loads A(6,4) with the 2-port description of the lumped-element model FET 1 of Figure 2. A 2-port description of the chip is assembled first without noise, noise parameters of type Nset = 4 are installed by SUB Nload, and the noisy chip is packaged by embedding it in the lossless network shown.

Circuit-element values for two FET's, available from SUBprograms in the FARAFT file, are listed. It may be noted that P(15) and P(20) are not used; the user may store other quantities here at his convenience.

It is clear that T_{min}, R_{opt}, and g_n must be non-negative if the four noise parameters are to provide a physical description of the Rothe and Dahlke† pair of noise sources. An upper bound on T_{min} is also required to prevent these sources from being more than 100% correlated*. If g_n were not in millimhos, this condition would be

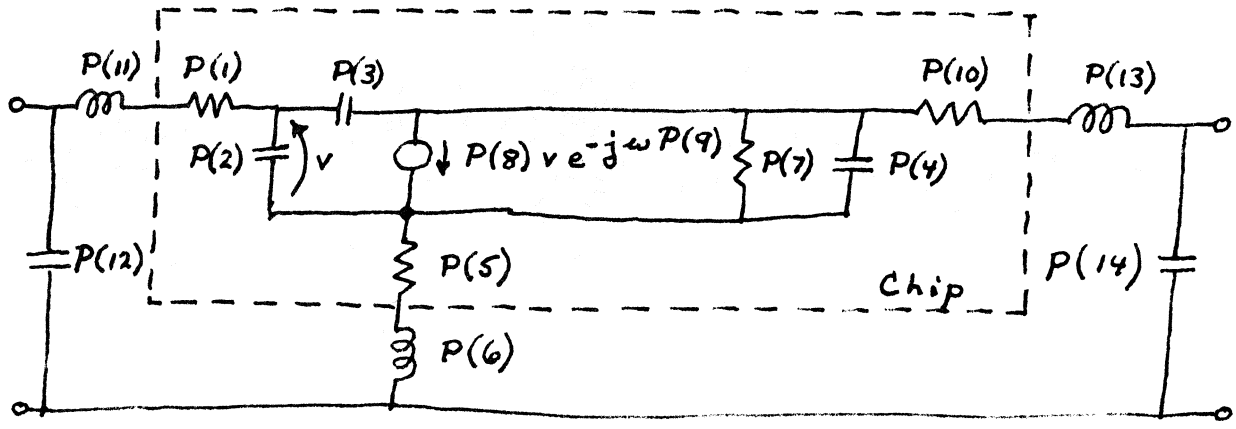
$$T_{min} \leq 4T_o R_{opt} g_n.$$

The corresponding condition on the circuit-element values reads

$$P(16) \leq 1.16P(17)P(19).$$

†H. Rothe and W. Dahlke, "Theory of Noisy Fourpoles," Proc. IRE, v. 44, pp. 811-818, 1956.

*Wojciech Wiatr, "A Method of Estimating Noise Parameters of Linear Microwave Two-Ports," Ph.D. Dissertation, Warsaw Technical University, Warsaw, Poland, 1980.



	P(n)	
n	SUB Ne673a	SUB Mgf 1412a
1	7.3	4
2	.42	.5
3	.0001	.031
4	.2	.35
5	1.55	2.3
6	.0006	.039
7	369	500
8	49	44
9	.3	5
10	1.6	3
11	.29	.4
12	.26	.25
13	.31	.35
14	.26	.45
15	0	0
16	13.9	19.3
17	207	187
18	805	240
19	.079	.092
20	0	0

Chip Noise Parameters:

$T_{min} = P(16) \times F$

$R_{opt} = P(17) / F$

$X_{opt} = P(18) / F$

$g_n = P(19) \times F^2$

Requirements for Physical Noise Sources:

$0 \leq P(17)$

$0 \leq P(19)$

$0 \leq P(16) \leq 1.16\, P(17)\, P(19)$

Figure 2. The Model FET 1.

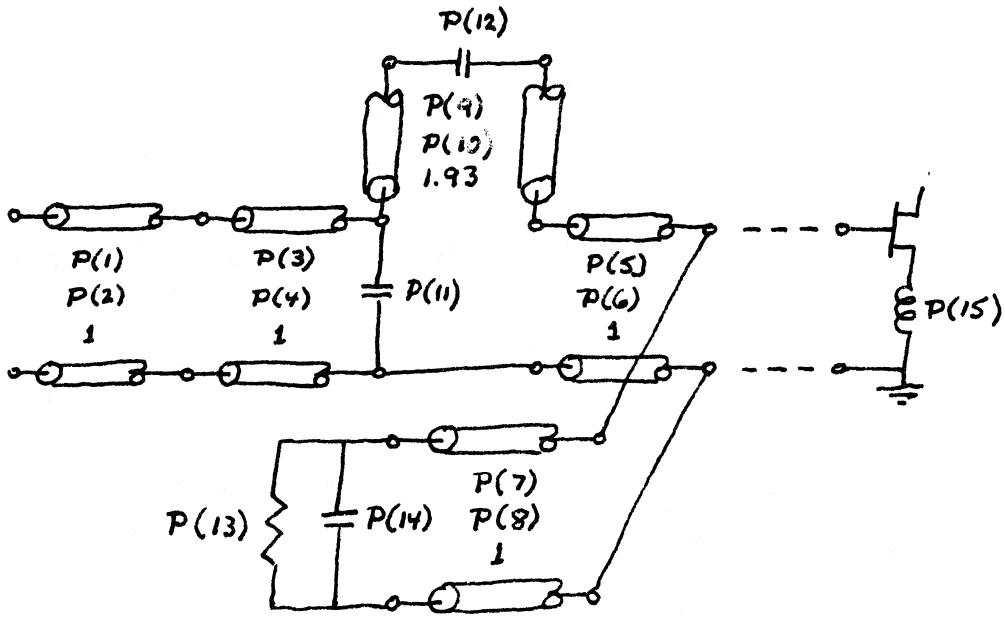
The chip noise parameters listed in Figure 2 are not those predicted by Pucel et al[#] because, except for g_n and X_{opt}, they depend differently on frequency. However, by a proper choice of P(16) through P(19), Pucel noise or the user's measured noise can be produced exactly at any single frequency. Since noise parameters do not vary wildly with frequency, a reasonable noise description can be expected for as much as an octave centered on the frequency where the noise parameters have been established.

In using SUB Optimize to adjust circuit-element values P(1) through P(14) so that the S-parameters of the model agree with FET measurements, time can be saved by omitting the calculation and loading of the chip noise parameters and by installing the resistance-inductance pairs [P(1),P(11)], [P(5),P(6)], and [P(10),P(13)] using only 3 CALLs of SUB Rlc, rather than the 6 CALLs required when the noise parameters are loaded. These shortcuts have been arranged by making SUB Fet1 responsive to the array element A(5,3) that is passed to it by the CALLing SUBprogram. If A(5,3) = 1, the shortcuts are used. In any case, A(5,3) is reset to zero by the first CALL of SUB Rlc.

SUB In1(A(*),P(*))

Using the values passed to it in P(15), this SUBprogram loads A(6,4) with the 2-port description of the FET input network of Figure 3, which has been found convenient for the work at the NRAO. The listed parameter values, from SUB K1 of the FARAFT file, are suitable for a K-band amplifier. The temperature of the gate bias resistance P(13) is taken to be 300K. Element P(15) stores

[#]R. A. Pucel, H. A. Haus, and H. Statz, "Signal and Noise Properties of Gallium Arsenide Microwave Field-Effect Transistors," Advances in Electronics and Electron Physics, v. 38, pp. 195-265, L. Marton Editor, Academic Press, New York, 1975.



From SUB K1 :

n	P(n)	Description
1	26	T_1
2	.126	
3	50	Tuner
4	.145	
5	100	Gate lead
6	.01	
7	90	Gate bias
8	.125	
9	39	Gate cap.
10	.075	
11	0	Discont. cap.
12	.04	End cap.
13	50	Bias R
14	.5	Bias C
15	.001	Source L

P(1) through P(10) are lossless transmission line parameters. For each line, the required 3 parameters are listed on the schematic in the following order :

Z_0

length (inches)

ϵ/ϵ_0

Figure 3. Input Network with K-band Parameters.

the source lead inductance of the following FET, but its value is currently not used by any of the FET_LIB SUBprograms.

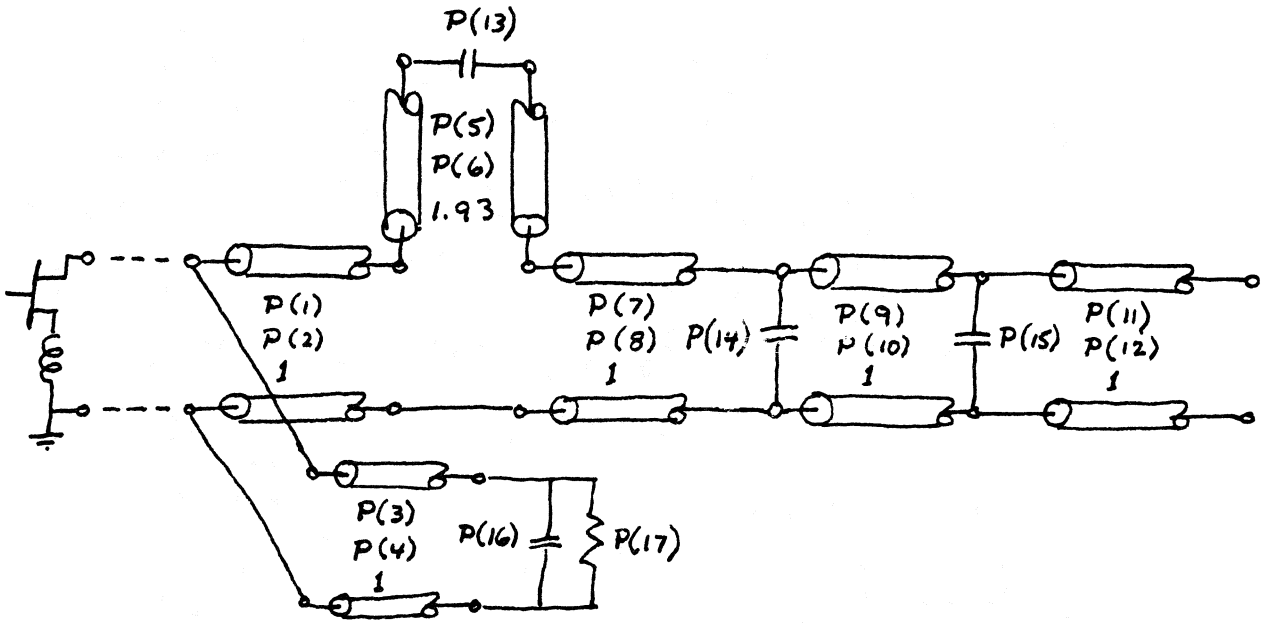
SUB Out1(A(*),P(*))

Using the values passed to it in P(17), this SUBprogram loads A(6,4) with the 2-port description of the FET output network of Figure 4. The listed parameter values, from SUB K2 of the FARAFT file, are suitable for a K-band amplifier. The temperature of the drain bias resistance, element P(17), is taken to be 300K. It may be noted that, because its length has been set to zero, the output line has currently been removed.

SUB Mplot(INTEGER Gmin,Gmax,Tmax,Opt)

This SUBprogram plots the decibel magnitudes of certain S-parameters of a 2-port against frequency, and it may also be instructed to plot the noise temperature of the 2-port. The plotted points are taken from the database, so it is appropriate to use SUB Saveckt to fill Dat(*). If noise temperature is also to be plotted, it should be installed in the database using SUB Nperformance following the CALL of SUB Saveckt.

S_{11} points are marked "I" as they are plotted to suggest "input return loss," and S_{22} points are marked "O" to suggest "output return loss." The scale of return gain extends from -40 dB to 0 dB. S_{21} points are marked "G" for "gain" and plotted on a scale extending from INTEGER Gmin dB to INTEGER Gmax dB. These three curves are always produced by SUB Mplot, regardless of the flag INTEGER Opt that it receives. If Opt = 0, the S_{12} plot is omitted, and the noise temperature of the 2-port, driven from 50 ohms, is plotted on a linear scale extending from zero to INTEGER Tmax. The plotted points are marked "T." But if Opt ≠ 0 -- Opt = 1 is suggested -- S_{12} is plotted instead, with points marked "R" to suggest



From SUB K2:

n	P(n)	Description
1	100	Drain lead
2	.01	
3	90	Drain bias
4	.125	
5	39	Drain cap.
6	.04	
7	50	Tuner
8	.01	
9	15	T_2
10	.126	
11	50	Out. line
12	0	
13	.04	End cap.
14	.03	Discont. cap.
15	.03	Discont. cap
16	.5	Bias C
17	50	Bias R

P(1) through P(12) are lossless transmission line parameters. For each line, the required 3 parameters are listed on the schematic in the following order:

Z_0

length (inches)

ϵ/ϵ_0

Figure 4. Output Network with K-band Parameters.

"reverse gain." The scale extending from Gmin dB to Gmax dB is also used for this reverse gain, but to keep the curve on the graph, what is actually plotted is

$$20 \log_{10}|S_{12}| + \text{Iso} \quad ; \quad \text{Iso} = \text{Tmax}.$$

The user must arrange to pass a value of this isolation parameter to SUB Mplot that is appropriate for his circuit. Tic marks divide the return loss scale into 8 intervals of 5 dB each and, regardless of the frequency range of the data stored in Dat(*), the frequency scale is divided into 10 intervals.

It is supposed that SUB Mplot will issue the first set of GRAPHICS commands of the current RUN of FARANT. Accordingly, SUB Mplot INITializes Graphics and turns the GRAPHICS screen ON; it clears the ALPHA screen, which will contain any printout directed to the CRT; and it SCRATCHes typing-aid KEY definitions and their labels, which are displayed on the GRAPHICS screen. When the graph has been PLOTted, LABELed, and Lettered by the user, the GRAPHICS screen is DUMPed to the printer and CLEARed, and the PRINTER IS reset to the device current when SUB Mplot was CALLed.

Unfortunately, HP9816 BASIC continues the mistake of HP9845 BASIC of leaving the pen down after LABELing. For this reason, the following plotting loop from SUB Mplot would produce unexpected marks under the T's if it were not for the PENUP statement:

```
FOR R = 1 TO Count
  PLOT Dat(R,1),Dat(R,14)
  LABEL "T"
  PENUP
  PLOT Dat(R,1),Dat(R,14)
NEXT R
```

SUB Web(A(*),INTEGER Opt)

This plot SUBprogram is intended to offer the user a picture of how the input impedance of his 2-port varies with the load impedance (when the flag INTEGER Opt = 1) or how the output impedance varies with source impedance (when Opt = 2). The 2-port description, at a single frequency, is passed to SUB Web in A(6,4). A(*) should contain a valid noise description so that SUB Web can mark with an "O" the optimum source impedance Z_{opt} (when Opt = 1) or the output impedance with the 2-port driven from Z_{opt} (when Opt = 2). The plots are made on the Smith chart, so it is reflection coefficients that are plotted, specifically the conjugates of reflection coefficients: $\Gamma_{in}^*(Z_l)$ when Opt = 1 and $\Gamma_{out}^*(Z_s)$ when Opt = 2. The points marked "O" are the optimum source reflection coefficient Γ_{opt} when Opt = 1 and $\Gamma_{out}^*(Z_{opt})$ when Opt = 2. In addition, the point $\Gamma_{in}^*(Z_o)$ is marked "*" when Opt = 1.

Six curves are plotted. The outermost is the circle obtained from

Γ_l or $\Gamma_s = e^{j\psi}$, ψ varied in 10-degree steps.

The five inner arcs result from load or source resistances of 0.2, 0.5, 1, 2, and 5 times Z_o, with the corresponding reactance varied through -5, -2, -1, 0, 1, 2, 5, and 5000 times Z_o. The user is invited to edit these resistance and reactance values and also the parameters used in CALLing SUB Smith to suit the 2-port being investigated.

2.3 Versions of FARAFT

The eight FARAFT SUBprograms listed in Figure 1 are contained in the FARAFT file and have been described in EDIR No. 217 and above. This file is offered as a framework that the user can modify and maintain as his own version of FARAFT. The other versions of FARAFT noted in Figure 1 and also STOREd on the FARAFT disc are discussed next; the user may also wish to appropriate one of these.

SLICE1

The SUBprograms in this file make INPUT and OUTPUT plots on the Smith chart as the frequency is varied. An amplifier consisting of an input network, a FET, and an output network is analyzed, for the INPUT plots, by SLICEing it at the junction of input network and FET, and for the OUTPUT plots, by SLICEing it between FET and output network. The INPUT curves are source reflection coefficient with the input network driven from 50 ohms, the conjugate of the input reflection coefficient of the FET and output network with a 50-ohm load, and the optimum source reflection coefficient for least noise. These plots are marked "S", "I*", and "Opt", respectively, at their low-frequency ends. The OUTPUT curves are load reflection coefficient with the output network terminated in 50 ohms and the conjugate of the output reflection coefficient of the FET and input network driven from 50 ohms. These plots are marked "L" and "O*", respectively, at their low-frequency ends. All curves but "Opt" are plots of the appropriate S-parameters. "Opt" is calculated from the noise parameters of the FET and output network, with a noiseless load assumed.

The points for all plots are stored in the database Dat(*) before plotting begins. Needless to say, these points are not stored in the standard form used by SUB Saveckt, SUB Nperformance, and SUB Prt. As written, SLICE1 CALLs FET data from SUB Ne673a and input and output network data from SUB K1 and SUB K2, so it is a K-band amplifier that is analyzed, from 21 to 25 GHz. But the user should edit SLICE1 to incorporate his data and his band of frequencies.

673A_FIT

The SUBprograms in this file adjust the circuit element values of the model FET 1, shown in Figure 2, so that its S-parameters agree as well as possible with measured S-parameters. The adjustment is made by SUB Optimize. Noise parameters are not used, and the input and output capacitances, P(12) and P(14),

are left as they were estimated when SUB Ne673a was written. So a 12-variable optimization is performed. The measured S-parameters could have been installed in SUB Pread and CALLed, but instead, it has been chosen to install them in DATA statements at the end of SUB Cktanalysis, together with a shortened version of the SUB Pread statements. NEC data, taken every gigahertz from 2 to 18 GHz, is used.

Initial values of the 12 variables are read into X(*) by SUB Farstart. Circuit element values are calculated by squaring these variables, so if a variable is driven negative in the course of optimization, the corresponding circuit element value remains positive. SUB Farstart now CALLs SUB Cktanalysis with X(*) and Opt = 2. Cued by Opt, SUB Cktanalysis CALLs SUB Ne673a to load Q(20) with FET circuit element values, of which only Q(12) and Q(14), the input and output package capacitances, will ever be used. The remaining 12 are then re-filled by squaring the 12 X(*) values. SUB Cktanalysis next READs the measured S-parameter data and stores it in S(17,8) in rectangular form for rapid manipulation. Before EXITing to SUB Farstart, a plot is made of each of the model's S-parameters, including measured points, and the initial circuit element values are printed.

Note that the parameters in S(*) and two of the values of Q(*) will be needed each time SUB Cktanalysis is CALLed by SUB Optimize for an objective function value. If these arrays were declared in a DIMension or an ALLOCATE statement, their values and memory locations would be discarded whenever SUB Cktanalysis is EXITed. A COMmon declaration would also have to appear in the mainline program and in each SUBprogram using COMmon. But a <u>labeled</u> COMmon declaration survives SUBEXIT or SUBEND and can be unique to just one SUBprogram.

```
COM /Ckta/ Q(20),S(17,8)
```

is used in SUB Cktanalysis to make Q(*) and S(*) available to subsequent CALLs.

The objective function calculated by SUB Cktanalysis is

$$\text{Fvalue} = \frac{1}{4 \times 17} \sum_{f=2}^{18} \sum_{i,k=1}^{2} \left| \frac{S_{ik}(f)_{\text{meas.}} - S_{ik}(f)_{\text{ckt.}}}{S_{ik}(f)_{\text{meas.}}} \right|^2 .$$

Dividing by the measured S-parameter values places as much emphasis on a small error in a small S-parameter value, like S_{12}, as on a large error in a large value, like S_{21}.

After the first SUBEXIT of SUB Cktanalysis, SUB Farstart CALLs SUB Optimize, passing X(*) and Dtype = 1 -- numerical derivatives. SUB Optimize is "in charge" from here almost to the end. It will CALL SUB Cktanalysis with Opt = Dtype = 1, expecting just Fvalue in RETURN, and cued by Opt, SUB Cktanalysis is arranged to produce Fvalue as quickly as possible. SUB Optimize prints its step-by-step partial results on the CRT. At the end it prints the number of Steps and the value of Eps, its "quit" parameter, followed by initial and final parameter values and objective function values. SUB Cktanalysis, CALLed with Opt = 0, prints final circuit element values and makes plots of the model's final S-parameters.

673A_MFIT

This file uses SUB Optimize to adjust 6 of the circuit element values of the FET 1 model in search of an agreement between the squared magnitudes of the circuit S-parameters and the corresponding measured squared magnitudes. The circuit is a K-band circuit consisting of the input network of SUB In1, the FET model of SUB Fet1, and the output network of SUB Out1, in cascade, with data, except for the adjusted values, taken from SUB K1, SUB Ne673a, and SUB K2 of the FARAFT file. The adjusted circuit element values, which are calculated

as the squares of the parameter values used by SUB Optimize and thus kept positive, are

P(3) = X(1)*X(1) = gate-drain capacitance

P(11) = X(2)*X(2) = gate lead inductance

P(6) = X(3)*X(3) = source lead inductance

P(13) = X(4)*X(4) = drain lead inductance

P(1) = X(5)*X(5) = gate series resistance

P(10) = X(6)*X(6) = drain series resistance

Measurements at only the three frequencies 21.6, 22.6, and 23.6 GHz are used. The objective function has been chosen to put more weight on errors in S_{11} and S_{22}. Using subscript c to mean an S-parameter calculated from the circuit and subscript m to mean a measured S-parameter,

$$\text{Fvalue} = \sum_{i=1}^{3} \left(|S_{12c}(f_i)|^2 - |S_{12m}(f_i)|^2 \right)^2 + \left(|S_{21c}(f_i)|^2 - |S_{21m}(f_i)|^2 \right)^2 + 10 \left[\left(|S_{11c}(f_i)|^2 - |S_{11m}(f_i)|^2 \right)^2 + \left(|S_{22c}(f_i)|^2 - |S_{22m}(f_i)|^2 \right)^2 \right]$$

Because S_{12} is small, its errors hardly contribute.

Like 673A_FIT, this file produces plots of initial S-parameters and a printout of initial circuit element values; the results of SUB Optimize are printed; and final circuit element values are printed and final S-parameter plots are produced. The only difference is that the plots are made by SUM Mplot with Opt = 1, so that the magnitudes of all four S-parameters are plotted against frequency.

In writing SUB Cktanalysis for this file, it has been convenient to use a labeled COMmon statement to retain some of the FET circuit element values and the input and output circuit descriptions. SUBroutines Pack and Unpack transfer ABCD descriptions of the input and output networks between A(6,4) and labeled COMmon.

ROSENBROCK

This file offers the user a quick test of SUB Optimize that he can re-RUN many times to familiarize himself with the properties of this SUBprogram. RUN time is less than 13 seconds. Rosenbrock's valley†, the objective function

$$\text{Fvalue} = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2,$$

is used. Clearly the minimum, Fvalue = 0, is reached at $x_1 = x_2 = 1$. The difficult starting point $x_1 = -1$, $x_2 = 10$ is used for each test. If the valley floor is taken to be the roughly parabolic strip within which the gradient direction and the direction <u>across</u> the valley are within 45 degrees of being perpendicular to each other, the width of the valley floor is only 0.00034 at the point $x_1 = -3.07$, $x_2 = 9.46$ where the floor is first reached by SUB Optimize. The maximum width of 0.021 occurs near $x_1 = x_2 = 0$, and the width drops to zero at the solution point. So this is a narrow, curved valley that offers severe optimization problems, in spite of the fact only two variables are used. Nearly 80 Steps of optimization are required.

The user is asked to INPUT the exponent of Eps, the SUB Optimize "quit" parameter, at the start of each RUN. He can change between the use of numerical

†H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," <u>The Computer Journal</u>, vol. 3 (1960), pp. 175-184.

and analytic derivatives that SUB Optimize will use in evaluating the gradient by EDITing the last line of PRINTed heading produced by SUB Farstart and, correspondingly, the parameter Dtype with which SUB Farstart CALLs SUB Optimize. No changes are required in SUB Optimize or in SUB Cktanalysis. When the calculations have been made and the RUN time PRINTed, the program is PAUSEd in SUB Farstart. From this point, pressing CONTinue will provide a heading for the next RUN, whereas just pressing RUN will avoid the heading.

The lessons from this file are as follows:

(1) Although the Davidon algorithm† implemented by SUB Optimize provides quadratic -- and hence rapid -- convergence toward the end, this desirable, speedy pursuit of a solution only persists for the last two or three Steps. In a lengthy problem like Rosenbrock's valley and most multivariable circuit problems, most of the Steps are taken in reaching a position from which quadratic convergence can begin.

(2) Increasing the SUB Optimize "quit" parameter Eps above its default values -- 10^{-15} for numerical derivatives and 10^{-30} for analytic derivatives -- merely eliminates some of the final Steps of quadratic convergence, reducing final accuracy without appreciably shortening RUN time. However, making Eps as large as 10^{-1} will stop optimization quickly, usually at the next Step.

(3) Reducing Eps below its default values may increase RUN time, but by and large without increasing final accuracy. No such reduction can lock SUB Optimize in an infinite loop.

---

†W. C. Davidon, "Optimally Conditioned Optimization Algorithms Without Line Searches," Math. Programming, vol. 9 (1975), pp. 1-30.

(4) Final accuracy is improved by using analytic derivatives. In ROSENBROCK, the exact solution point, Fvalue = 0, is reached when they are used, but not when numerical derivatives are used.

3.0 Translations from HP9845 BASIC to HP9816 BASIC

Much of HP9845 BASIC is compatible with HP9816 BASIC. Both languages recognize the mainline program, one of its SUBprograms, and one of its multiple-line DEFined FunctioNs as distinct contexts. If it is not passed between two contexts as a pass parameter or declared in a COMmon statement contained in both contexts, a single or array, variable or string can be manipulated in one context without having any effect on its value in the other context, in spite of the fact that the same name is used in both contexts.

A variable and a string having the same name -- X and X$, for example -- can coexist in one context in either language. In HP9845 BASIC, a single variable X and an array variable X(*) of the same name can even coexist in one context, but HP9816 BASIC does not allow this. One of the major tasks in translating FARANT from HP9845 BASIC to HP9816 BASIC was renaming variables as required to ensure that single variables and array variables in each context all have distinct names. Another task is modifying the GRAPHICS statements, several of which are written differently in the two languages. HP9816 BASIC offers several statements for dealing with arrays that are not available in HP9845 BASIC and that the user may wish to take advantage of. Conversely, a number of statements -- including LETTER -- that were available in HP9845 BASIC are not available in HP9816 BASIC.

Hewlett-Packard offers a Series 200 BASIC Translator, a disc containing software that can translate programs from HP9845 BASIC to HP9816 BASIC. HP9845 programs that have been SAVEd on tape are transferred to disc on the HP9121 dual disc drive under control of the HP9816 computer, with no need to retype.

The manual for the Translator, particularly its Chapter 6, should be consulted for a discussion of the differences between the two languages.

Unfortunately, the Translator translates into BASIC 2.0, the operating system provided by just the BASIC System disc. It does not translate into Extended BASIC 2.0 or Extended BASIC 2.1, which becomes the operating system after one of the binary files AP2_0 or AP2_1 is LOADed. The user should hence not be surprised to find remarks in the manual like "The MAT family of statements is not supported [...], and all MAT statements will be [converted into comments] by the translator with a warning message." Evidently BASIC 2.0 also does not support the statement

```
Y = MAX(X1,X2,X3),
```

either, because the Translator will convert this statement into a multiple-line DEFined FunctioN and attach its code at the end of the program, even though that statement <u>is</u> supported by Extended BASIC. For some reason, the Translator does not recognize different contexts in the code to be translated. If X appears in one context and X(*) in another, the translated X(*) will be renamed X_(*). For this reason it is wise to translate only one context at a time. At best, then, the Translator is a device for warning the user unfamiliar with HP9816 BASIC of the portions of his code that need attention.